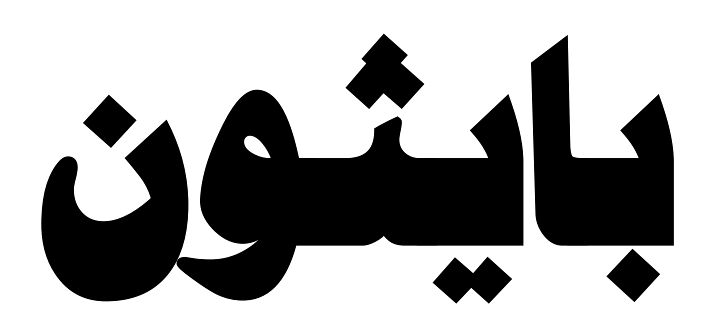


بايثون
للجميع
المستوى الأول
تاليف واعداد
م / حسن على

تعلم أساسيات لغة بايثون بكل سهوله



للجميع

المستوى الأول

تأليف وأعدد

م / حسن على

الطبعة الأولي

الكويت - 2019

إِذا ما أَقامَ العلمُ رايةَ أمةٍ

فليس لها حتى القيامةِ ناكسُ

الطبعة الأولى، الكويت/ 2019
المؤلف: م/حسن علي

رقم إيداع:

مقاس الكتاب
17*24 cm
الغلاف: المؤلف
الناشر : المؤلف




بسم الله الرحمن الرحيم

الإهداء

إلى من ساندوني وعلموني من أعلمهم وهم لا يعلموني إلى أولئك الذين أثروا الساحة العربية بشروحات تعليم البرمجيات.

. إلى من اشتري لي أول جهاز كمبيوتر من ماله فكانت الانطلاقة، إلي أبي العزيز أسال اللّه أن يتغمدك برحمته وان نكون معك شركاء في اجر هذا الكتاب.

إلى من يمدون يد العون لغيرهم.

اليكم جميعاً اهدي هذا الكتاب

قبل أن تقرا هذا الكتاب

هذا الكتاب يقدم لك شرح أساسيات لغة البايثون بطريقة سهلة وبسيطة فتعلُم أي لغة برمجة لا يتوقف على كتاب أو على طريقة عرض إنما يتوقف عليك أنت فإليك بعض النصائح لمشكلات قد تواجهك أثناء قراءاتك لهذا الكتاب.

- **حدد هدفك**

إن لتحديد الهدف دور كبير في تنفيذه كن واضح الرؤية مُحدد الهدف **قل إن هدفي هو تعلم لغة البايثون.**

- **حدد الوقت**

الازم لتحقيق هذا الهدف لا يمكن لهدف أن يتحقق مالم يوضع في إطار زمني محدد لا يعني ذلك أن تأتي بجدول زمني بالأيام والساعات ولكن قل لن يأتي شهر كذا إلا وانا أجُيد لغة البايثون.

- **اصبر وثابر**

كمبتدئ في عالم البرمجة ستواجهك عقبات كثيرة على هذا الطريق قد تقضي الساعات لفهم جزئية بسيطة وقد يمر عليك اليوم ولم تفهم شيء فاصبر على التعلم وعاود المحاولة مرات ومرات إياك أن تيأس.

- **لا تنوع المصادر**

ففضاء الأنترنت مليء بمصادر تعلم بايثون لا تتنقل من مصدر إلى أخر قبل أن تنهي الأول. جميع المصادر يشرحون نفس اللغة قد تختلف طريقة الشرح قليلا فلا تشتت نفسك بين المصادر.

- **لا تبنى قبل أن تتعلم البناء**

لا تنتقل من مستوي إلى مستوي اعلى قبل أن تلم بالمستوي السابق فاذا أردت بناء بيت فيجب عليك تعلم البناء أولاً.
لم يولد أحد مبرمجا ولكن بالتعلم أصبح

- **لا تنتظر الثراء**

لتكن رغبتك في تعلم البرمجة منزهة من أي غرض مادي أو اجتماعي فالبرمجة ليست جني المصباح إنما تحتاج إلى مزيد من العرق والجهد والإرهاق فالمبرمج الناجح هو من يحب البرمجة لذاتها لا لغرض أخر.

- **لا تنظر إلى المغريات**

ستواجهك كثيراً من المغريات لتصرفك عن هدفك كان يجبرك صديق إن البايثون غير مجدية أو هناك مجالات أفضل لا تلتفت إليهم امضي في طريقك وكن مؤمن بهدفك.

- **وصفة لعلاج الإحباط والملل**

ستواجهك مشاكل لا تستطيع حلها تصل بك إلي درجة فقدان الثقة بنفسك لدرجة أن تقول من المستحيل أن أصبح مبرمج لو وصلت لتلك الحالة اغلق الكتاب وافعل أي شيء أخر اقرأ قران صلي اخرج مع أصدقائك اقضي وقت مع عائلتك ... الخ فبعد أن يصفي ذهنك افتح الكتاب من جديد وابدأ من حيث توقفت ستتمكن من حل المشكلة وستعلم كم كنت غبياً.

- **أنت رقيب نفسك**

لا تنتظر من أحد أن يتابعك أو يذكرك بهدفك إن لم تكن رقيب نفسك اترك هذا الكتاب وافعل أي شيء أخر.

- **لا تنظر بعينيك فقط**

لا تحاول تعلم البرمجة بمجرد النظر طبق عمليا اعد كتابة الكود بيدك وتعلم من أخطاءك وحاول التعديل على الكود إن كان هناك مجال وأعطي لنفسك امثله ونفذها.

- **إنسي مفتاحي ctrl+c**

لا تعتاد النسخ دع أصابعك تعتاد كتابة الكود.

- **لا تدع غيرك يفكر لك**

أثناء كتابتك للكود ستواجهك الكثير من الأخطاء نشط عقلك وابحث عن المشكلة ولا تلجا لغيرك لحلها إلا أذا عجزت عن ذلك

- **اترك من لا يؤمنون بك**

سيحاولون إحباطك والتهكم عليك لا تنظر إلى اولائك المخذلين امضي في طريقك على ثقة وإيمان بهدفك

-

في هذا الكتاب

سنتعلم سويا تهيئة بيئة العمل بتثبيت محرك لغة بايثون language compile وبرنامج IDE وقد اعتمدنا في شرحنا هنا برنامج eclipse لمجانتيه وسهولة العمل عليه ويمكنك العمل به على أكثر من لغة والكثير.....
وستتعلم أيضا أساسيات لغة البايثون وكيف تكتب كود برمجي بها وطباعة مخرجاتك في أل .console
كما ستعلم ماهي المصفوفات ال arrays وستتعرف على أنواع البيانات كالعددي والعشري والنصي والقائمة والقاموس وكيفية التعامل مع القوائم.

وسنتعرف على التكرار والشروط وتداخل الشروط واستخدام شرط داخل شرط وستلم بالدوال كيفية إنشائها ومتغيراتها وكيفية استدعائها.
وسنقلي نظرة على المكتبات وفكرة عامة عنها وكذلك المديول وكيف تنشئ مديولك الخاص.
وستتعلم كيفية إنشاء ملف وقراءة محتويات ملف وحذف ملف وإنشاء مجلد جديد.
وكيفية تخطي الأخطاء المتعلقة بالنظام وكيف تتعامل مع الوقت باستخدام مكتبةdatetime من حسابك لتاريخ اليوم أو تحديد تاريخ معين أو المقارنة بين التواريخ .... الخ.

كما سننتقل بك إلي التطبيق العملي لإنشاء برنامج أله حاسبة لحساب الزكاة الواجبة وسنتطرق إلى كيف يتم إعداد البرنامج من مرحلة التحليل إلى التصميم وتنفيذ الكود البرمجي.
ستعلم ماهية المكتبات باستخدام مكتبة tkinterالخاصة بالوجاهات الرسومية GUI وستتعلم كيفية إنشاء نافذة برنامج وإضافة الأزرار إليها وإضافة الaction الخاص بكل زر

يأتيك هذا الكتاب في فصلين الفصل الأول وهو خاص بأساسيات لغة البايثون والفصل الثاني التطبيق العملي على ما درسناه في الفصل الأول متمثل في برنامج حاسبة الزكاة.

تعرف على بايثون

بايثون هي لغة برمجة عالية المستوى تم إنشاؤها بواسطة Guido van Rossum. لديها بنية بسيطة سهلة الاستخدام، مما يجعلها لغة مثالية لشخص يحاول تعلم برمجة الكمبيوتر لأول مرة.

قبل أن تتعلم بايثون تعرف على تاريخها.

بايثون هي لغة برمجه مفتوحة المصدر، تم إنشاء الكثير من المكتبات والبرامج والتطبيقات المبنية عليها مثل (Django and Bottle and SymPy and NumPy) وكثير من GUI (واجهات المستخدم الرسومية) مثل (Pygame, Panda3D, Tinkerer)

متي بدأت بايثون؟

هي لغة قديمة أنشئها Guido Van Rossum بداء العمل عليها في أواخر الثمانينات وتم إصدار أول نسخة منها سنة 1994

لماذا تم إنشاء بايثون؟

في أواخر الثمانينات، كان Guido Van Rossum يعمل في Amoeba distributed أراد استخدام لغة سهلة وبسيطة (بنية بسيطة سهلة الفهم) لذلك، قرر إنشاء لغة سهلة التعلم. هذا أدى إلى تصميم لغة جديدة سميت فيما بعد بايثون.

لماذا سميت بايثون؟

لم تسمي على اسم الثعبان الخطير بايثون ولكن Rossum كان من معجبين بسلسلة " Monty Python's Flying Circus "

بايثون عبر التاريخ:

1. 1994
   - الإصدار 1.0
2. 2000
   - الإصدار 1.6
   - الإصدار
   - 2.0
3. 2010
   - الإصدار2.7
4. 2008
   - الإصدار3.0
5. 2015
   - الإصدار3.5
6. 2016
   - الإصدار3.5.2
7. 2017
   - الإصدار35.3
8. 2018
   - الإصدار3.5.5
9. 20-10-2018
   - الإصدار3.7.1

ما يميز بايثون:

1. بسيطة سهل التعلم.
2. مجانية ومفتوحة المصدر.
3. تعمل على اكثرمن نظام تشغيل مثل ويندوز وماك ... الخ.
4. ذكية في التعامل مع الهارد وير.
5. دعم كبير من المكتبات التي تؤدي مهام ووظائف مختلفة.
6. تدعم Object-oriented

الفصل الأول أساسيات لغة بايثون

في هذا الفصل سنتعلم سويا أساسيات لغة البايثون كـ تثبيت لغة البايثون وتثبيت برنامج **eclipse**

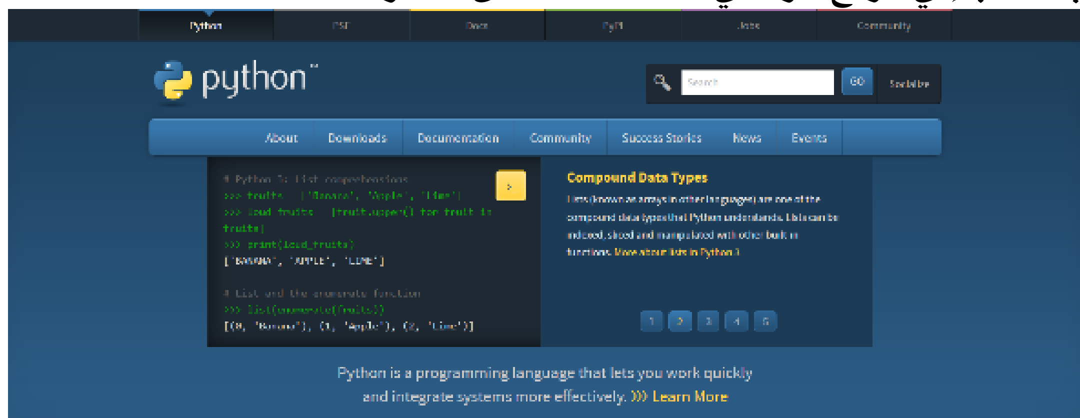
1) المتغيرات
2) التعليقات
3) الثوابت
4) التكرار
5) الدوال
6) أنواع البيانات
7) المصفوفات
8) الشروط
9) تضمين الملفات (الاستدعاء)
10) إدخال البيانات (طلب بيانات من المستخدمين)
11) طباعة البيانات (الإخراج)
12) تفادي الأخطاء **try & except**
13) التعامل مع الملفات
14) التعامل مع الوقت ومكتبة **datetime**

في نهاية هذ الفصل ستكون قادرا على فهم أي شفرة برمجية (كود) بايثون وتنفيذ برامج بسيطة بها.

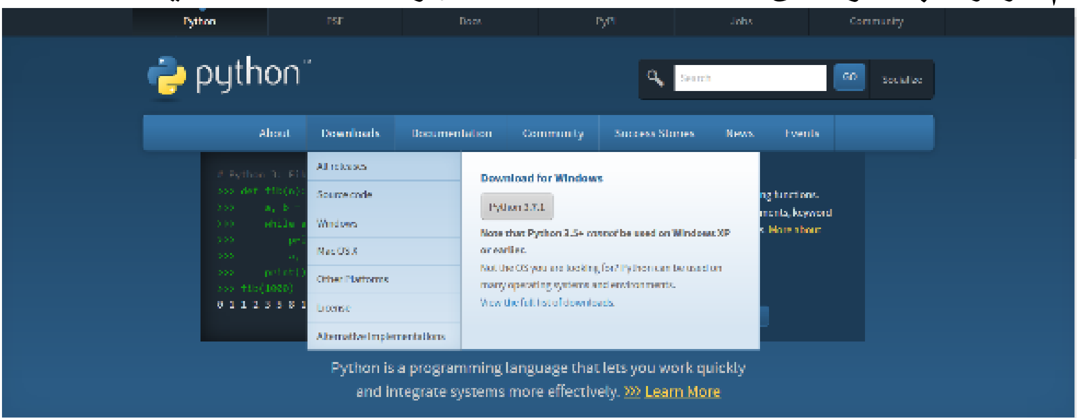
تثبيت بايثون – Python

تختلف طريقة تثبت اللغة من نظام تشغيل لأخر، في مثالنا التالي نستخدم نظام تشغيل ويندوز 8 كما تختلف طريقة التحميل حسب نوع المتصفح في مثالنا نستخدم متصفح فايرفوكس 63.0.3

بالذهاب إلي الموقع الرسمي للغة python على الأنترنت**https://www.python.org**



ثم الوقوف بالفارة على كلمة **Downloads** ستظهر لك النافذة التالية

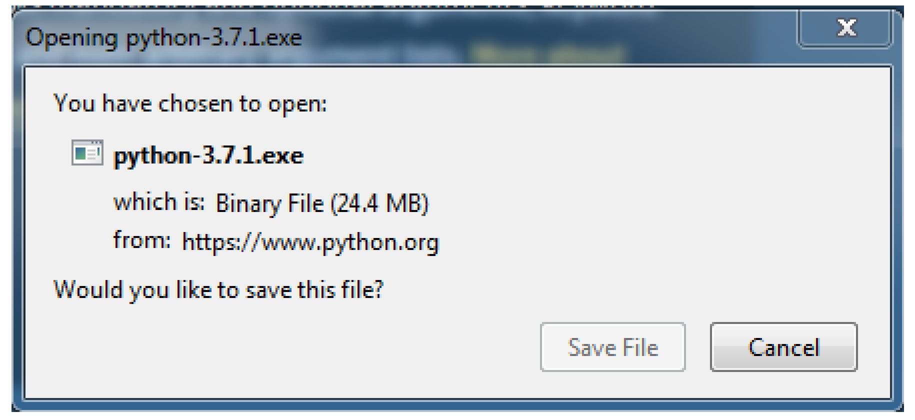


بإمكانك الاختيار بين أنظمة التشغيل بما يناسبك من الجزء الأيسر من القائمة (يقوم الموقع بتحديد النسخة المناسبة لك تلقائيا بناءا على مواصفات جهازك) أو بالضغط على الزر الرصاصي بالجانب الأيمن.

Python 3.7.1

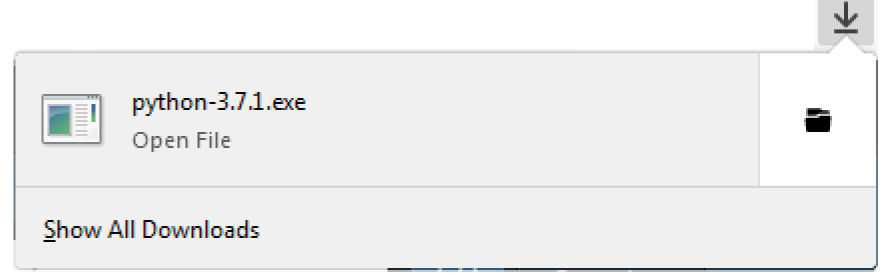
3.7.1 رقم إصدار لغة البايثون

ستظهر لك نافذة التحميل

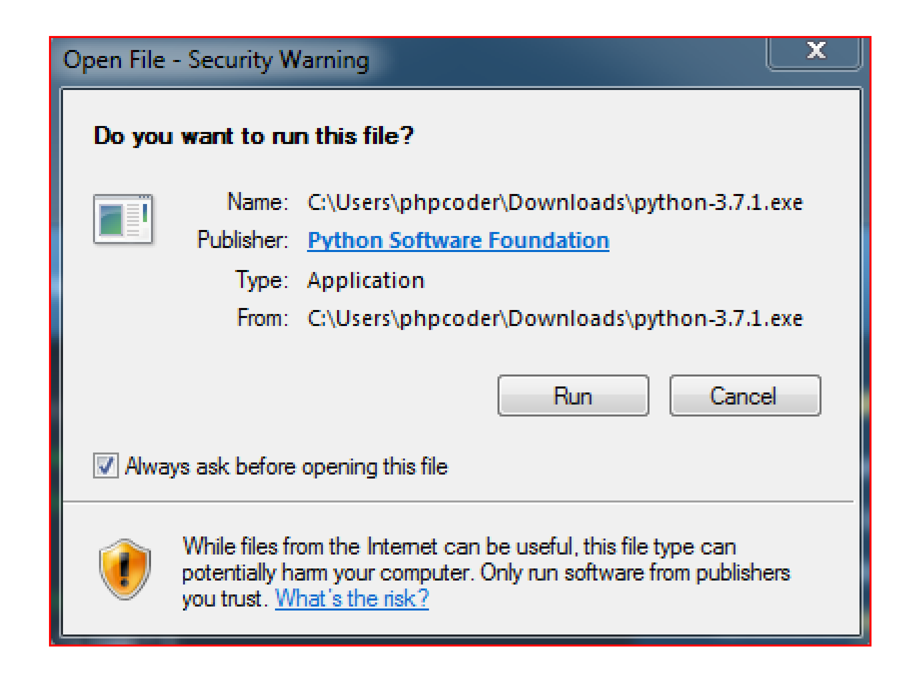


بالضغط على **save File** سيتم حفظ الملف في مجلد **Downloads** وإضافته إلى قائمة تحميلات متصفحك

اذهب إلى قائمة تحميلات متصفحك



بالضغط على اسم الملف **python-3.7.1.exe** سيظهر لنا نافذة تأكيد التثبيت



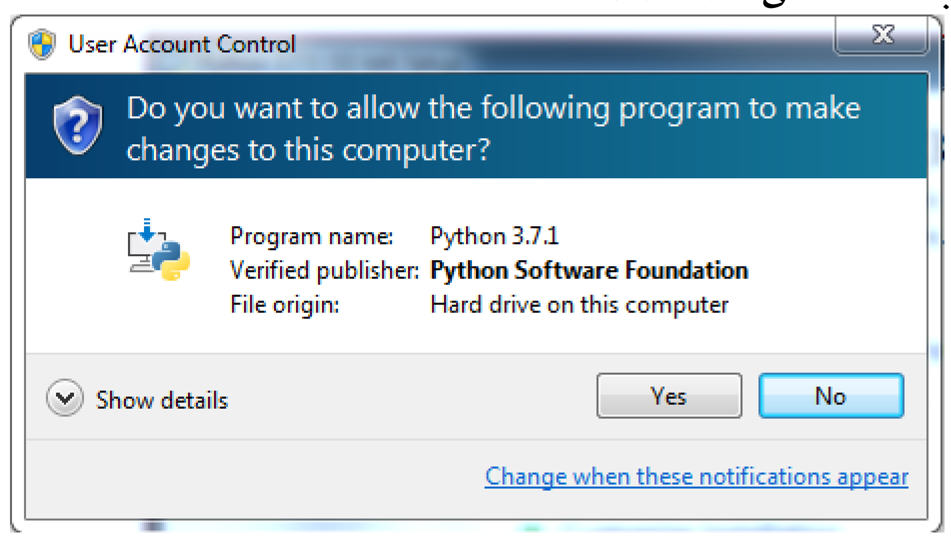
هذه النافذة قد لا تظهر في بعض المتصفحات فهي رسالة تحذيرية بان الملفات المحملة من الأنترنت عادة ليست امنه

ثم نضغط على Run

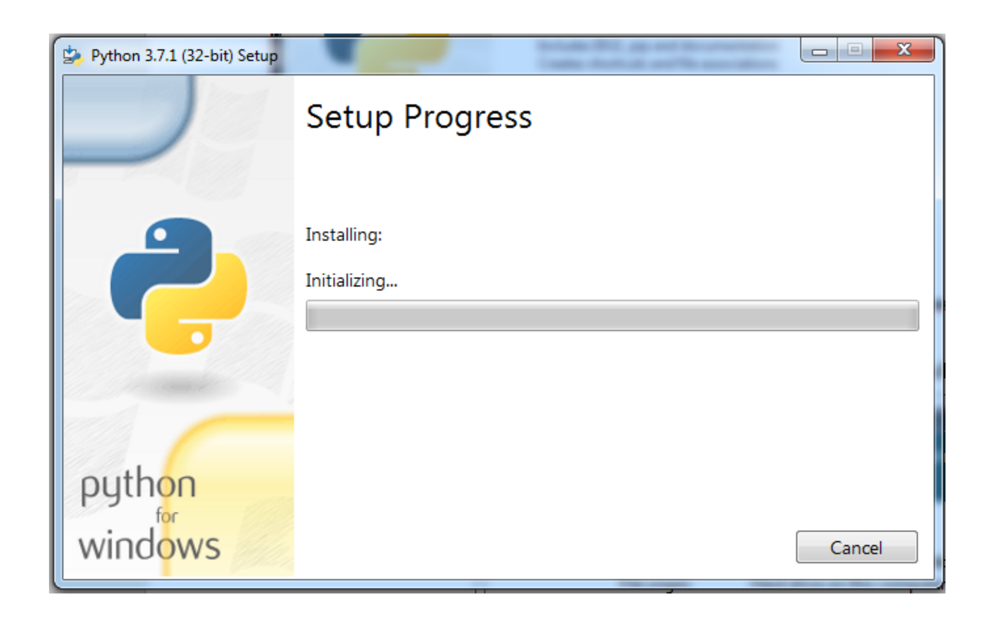
ستظهر لنا نافذة التثبيت الأساسية



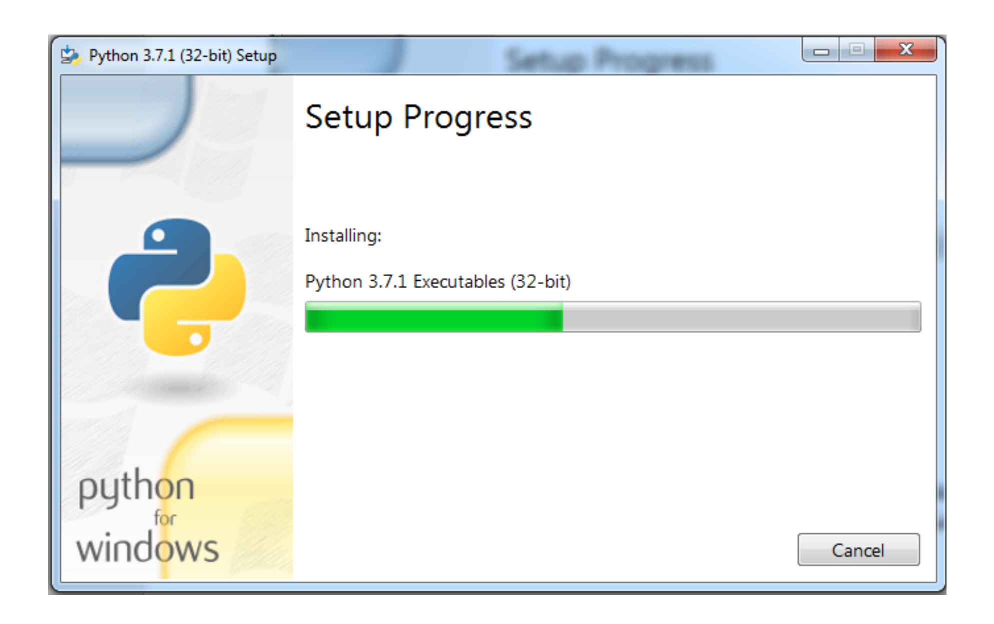
بالضغط على Install Now

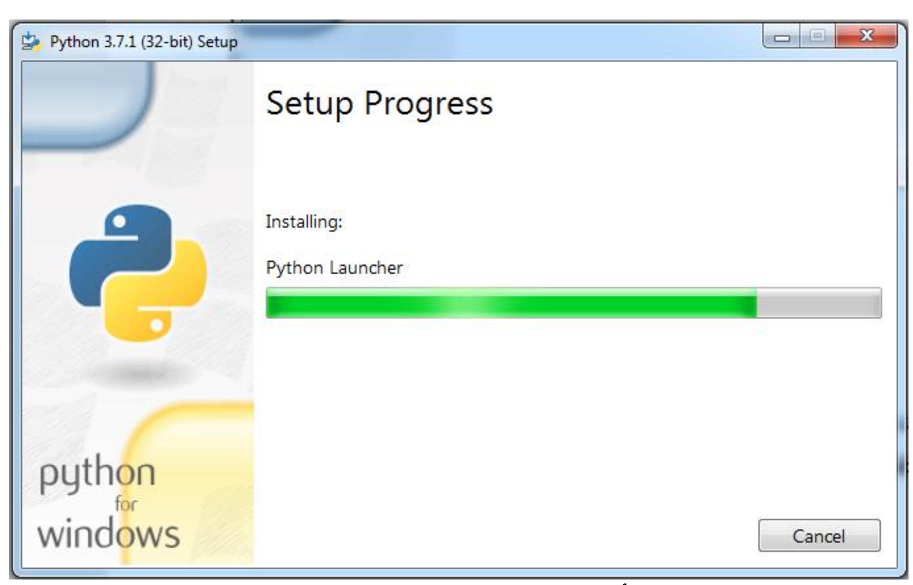


ثم بالضغط على Yes



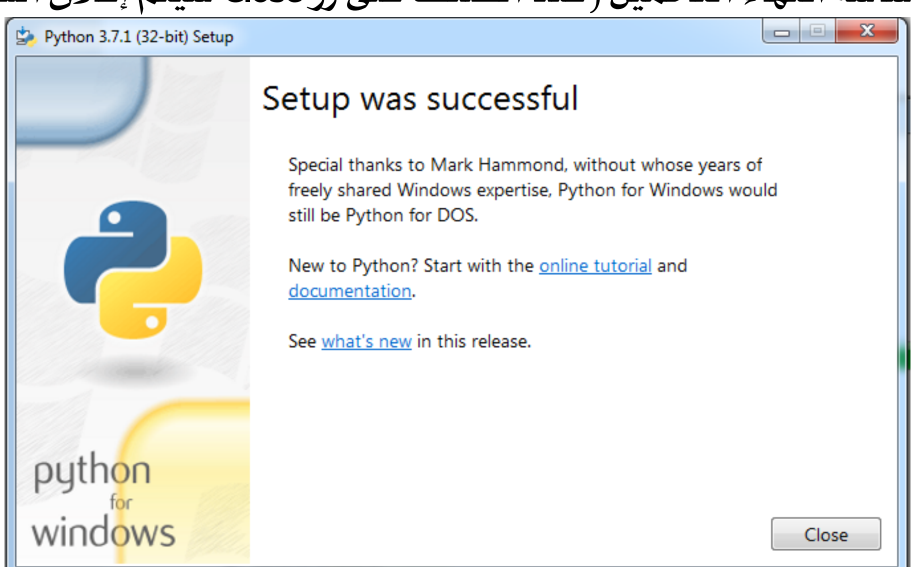
ننتظر حتى تنتهي عملية التثبيت





لاحظ زيادة المؤشر الأخضر الذي يشير إلى مرحلة تثبيت البرنامج

شاشة انتهاء التحميل (عند الضغط على زر **Close** سيتم إغلاق الشاشة)





مبروك لديك الأن محرك لغة Python على جهازك

نذهب إلى قائمة ابدأ (**start**) من الويندوز ونبحث عن كلمة **python** للتأكد من تثبيت اللغة على الجهاز

ثم نختار **IDLE** (وهو محرر لغة بايثون الاساسي) ستظهر لك نافذة المحرر

اكتب جملة I am python

واضغط على زر enter من لوحة المفاتيح هل تلاحظ طباعة كلمة I am python باللون الأزرق

```
Python 3.7.1 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Inte
l)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("I am python")
I am python
>>> |
                                                                    Ln: 5  Col: 4
```



تثبيت برنامج – Eclipse

سنقوم بتثبيت الإصدار رقم 4.9.0 على نظام التشغيل ويندوز

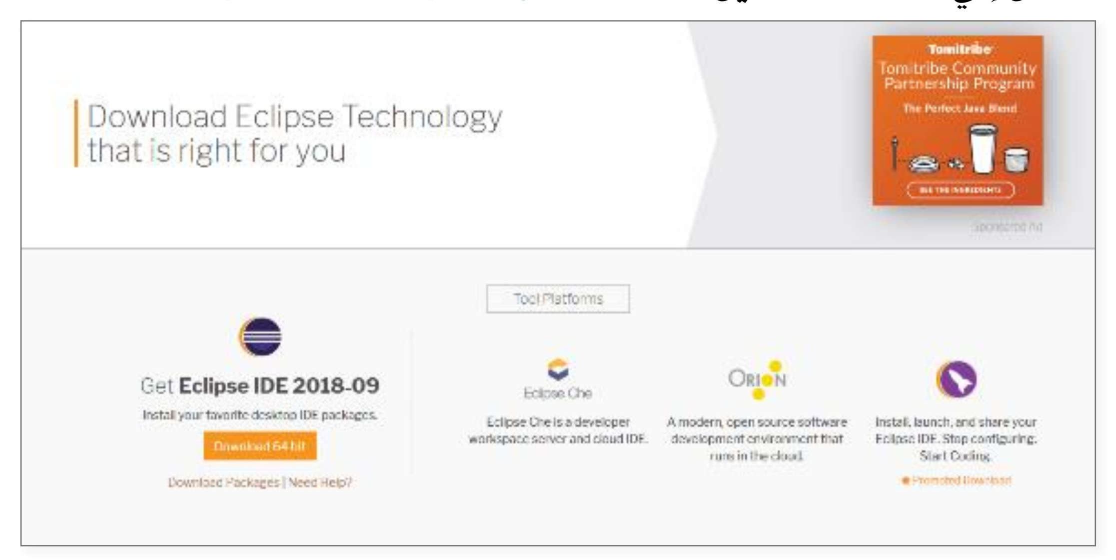
نذهب إلي الموقع الرسمي للبرنامج على الأنترنت https://www.eclipse.org



بالضغط على زر Download باللون البرتقالي اعلى يمين الصورة



سننتقل إلي صفحة التحميل https://www.eclipse.org/downloads

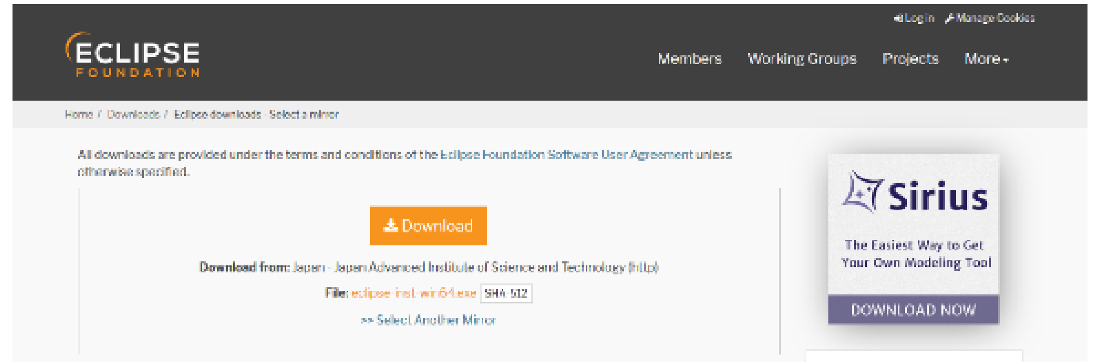


بالضغط على الزر البرتقالي Download 64 bit

Download 64 bit

46 bit هي نوع النواة للبروسيسور الخاص بالجهاز حيث يقوم الموقع بتحديد البرنامج بما يتوافق مع جهازك مثل نوع نظام التشغيل المستخدم أو حجم النواة

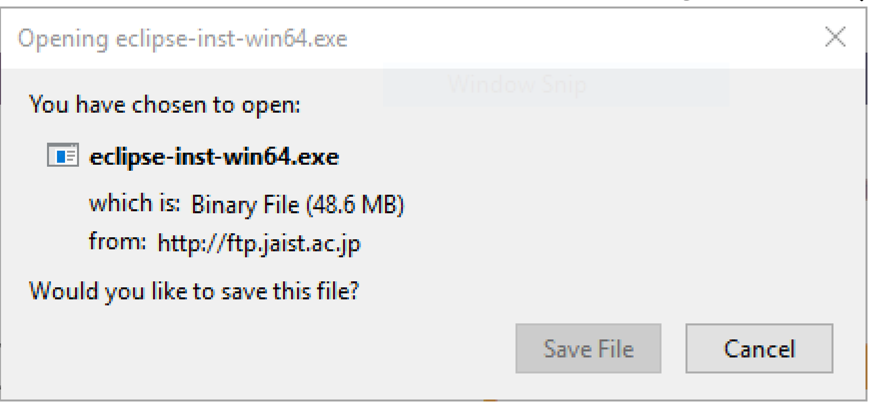
ننتقل إلى صفحة التحميل



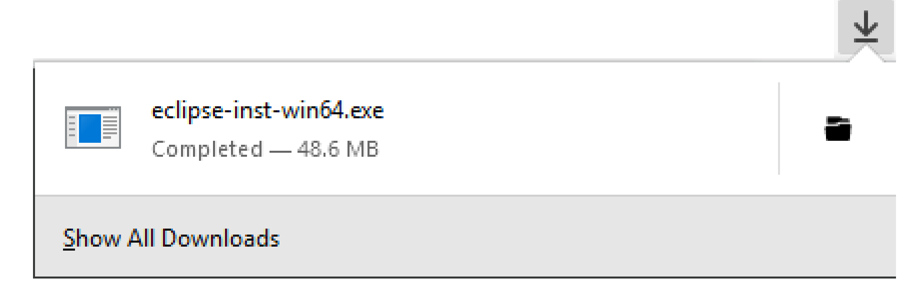
بالضغط على زر Download



بالضغط على Save File



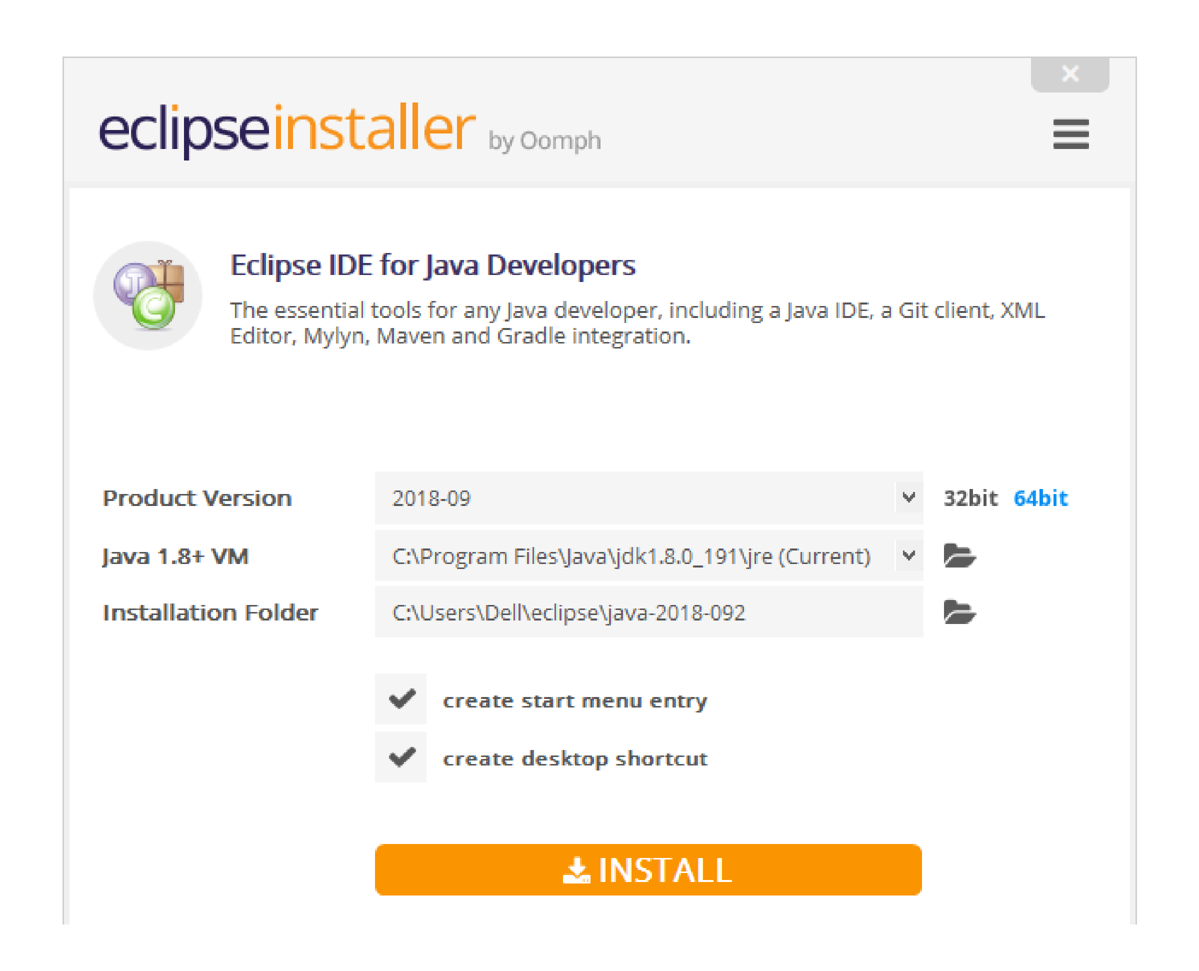
نضغط على اسم البرنامج من قائمة تحميلات المتصفح أو بالذهاب إلى مجلد **Downloads**



يتطلب برنامج **eclipse** مكتبة **Java SE Development Kit** (JDK) إن لم تكن مثبتة على جهازك قم بتحميلها وتثبيتها أولا بالدخول إلي هذا الرابط
https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

قم باختيار **Eclipse IDE for Java Developers**





ثم **INSTALL**

بدء عملية تثبيت البرنامج على جهازك

eclipseinstaller by Oomph

Eclipse IDE for Java Developers

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration.

Product Version	2018-09	64bit
Java 1.8+ VM	C:\Program Files\Java\jdk1.8.0_191\jre (Current)	
Installation Folder	C:\Users\Dell\eclipse\java-2018-092	

✔ create start menu entry

✔ create desktop shortcut

INSTALLING

✖ Cancel Installation

< BACK

eclipseinstaller by Oomph

Eclipse IDE for Java Developers

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration.

Product Version	2018-09	64bit
Java 1.8+ VM	C:\Program Files\Java\jdk1.8.0_191\jre (Current)	
Installation Folder	C:\Users\Dell\eclipse\java-2018-092	

- ✔ create start menu entry
- ✔ create desktop shortcut

▶ LAUNCH

show readme file

open in system explorer

BACK



مبروك أنت الأن لديك برنامج Eclipse على جهازك

اضغط على الزر الأخضر LAUNCH لفتح نافذة البرنامج

اختيار منطقة العمل (يقصد بمنطقة العمل هي المجلد الذي سيحتفظ البرنامج بملفات المشاريع المستقبلية فيه)

Eclipse IDE Launcher

Select a directory as workspace

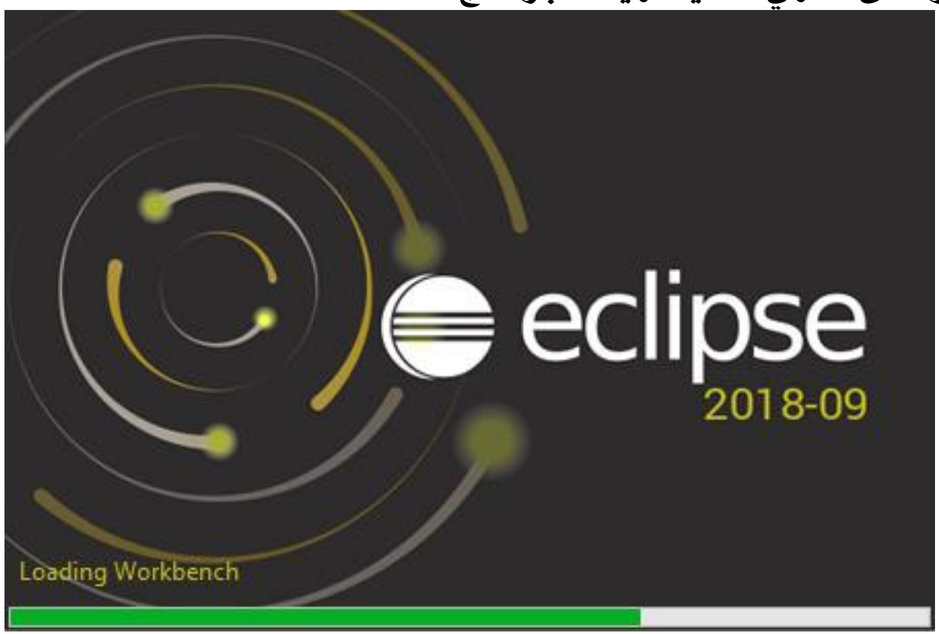
Eclipse IDE uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\Dell\eclipse-workspace   Browse...

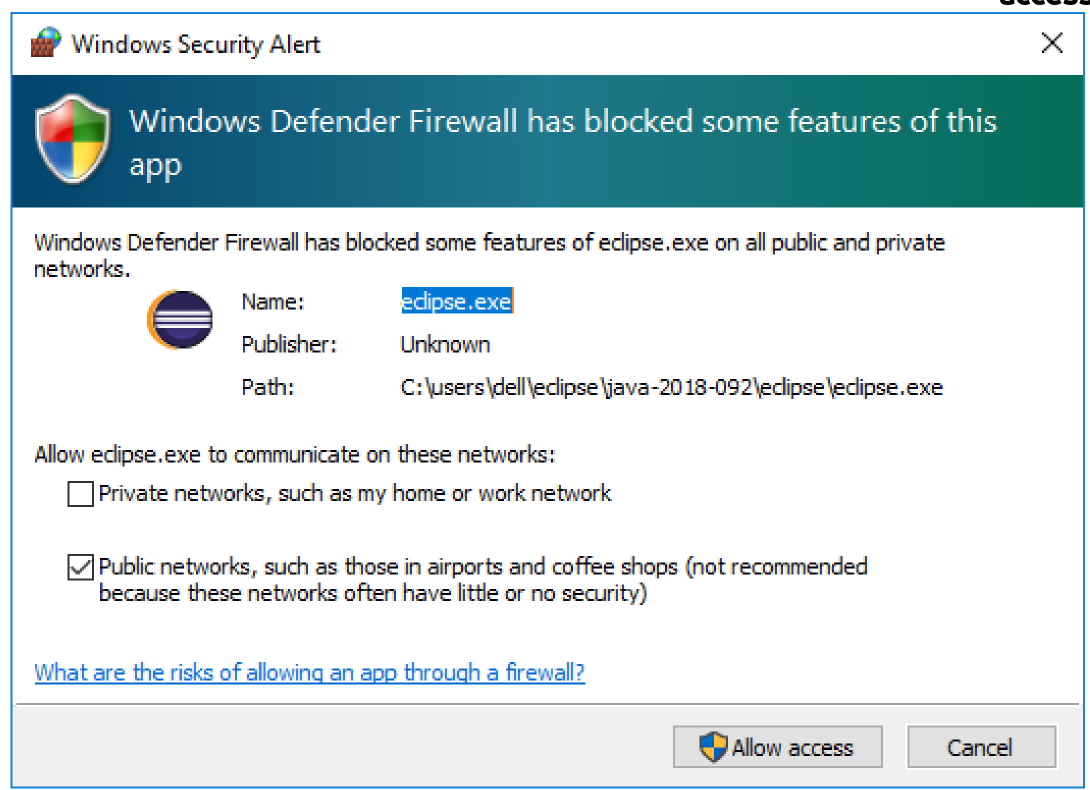
☐ Use this as the default and do not ask again

Launch   Cancel

انتظر حتى تنتهي عملية تهيئة البرنامج



قد تظهر لك هذه النافذة وهي خاصة بأمان نسخة الويندوز اضغط على زر Allow access



تركيب إضافة PyDev على برنامج Eclipse

من قائمة Help Marketplace نختار Eclipse

Help -> Eclipse Marketplace

اتبع الخطوات كما بالصورة

Eclipse Marketplace

Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

Search | Recent | Popular | Favorites | Installed | Eclipse Newsletter: Boot & Build Eclip...

Find: PyDev ❶ | All Markets | All Categories | Go ❷

PyDev - Python IDE for Eclipse 7.0.3

PyDev is a plugin that enables Eclipse to be used as a Python IDE (supporting also Jython and IronPython). It uses advanced type inference techniques which allow... more info

by Brainwy Software, EPL

IDE Python Aptana Pydev Django ...

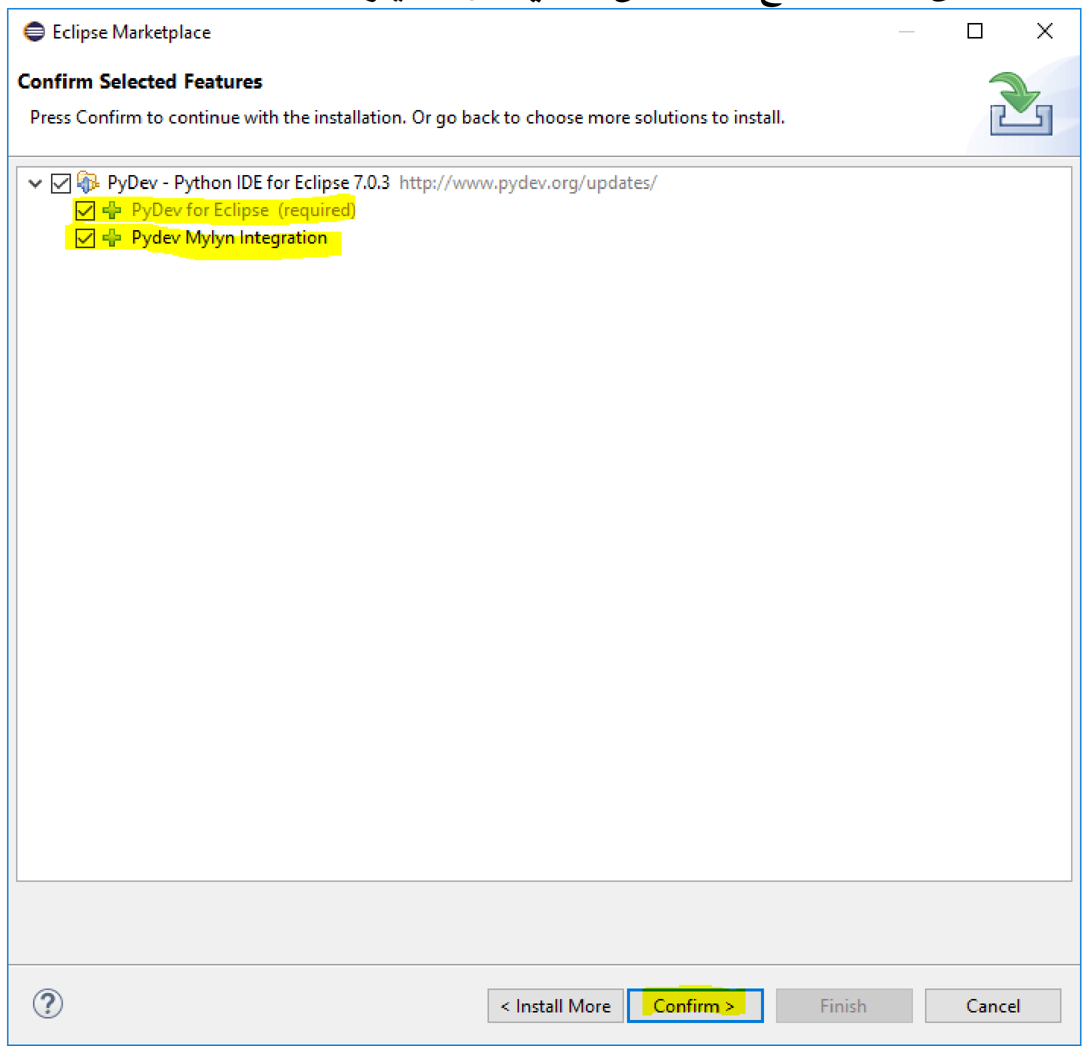
1302 | Installs: **1.07M** (19,272 last month) | ❸ Install

Marketplaces

< Back | Install Now > | Finish | Cancel

1. اكتب اسم الإضافة PyDev
2. اضغط على زر Go
3. اضغط على زر install

اضغط على confirm مع التأكد من تحديد كل الخيارات



Eclipse Marketplace

Review Licenses

Licenses must be reviewed and accepted before the software can be installed.

Licenses:

- Eclipse Public License
- Eclipse Public License - v 1.0

License text:

Eclipse Public License

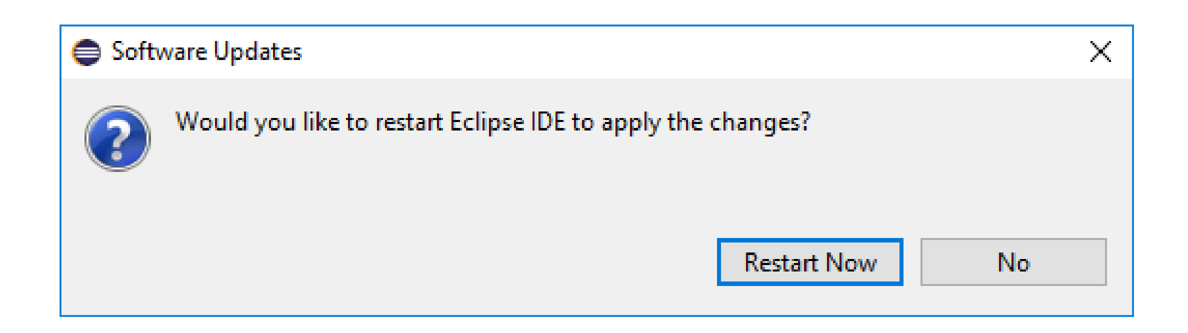
(•) I accept the terms of the license agreements

( ) I do not accept the terms of the license agreements

< Back | Next > | Finish | Cancel

قم بتفعيل Accept ثم اضغط على زر Finish

سيطلب منك البرنامج إعادة تشغيله للانتهاء من تثبيت PyDev

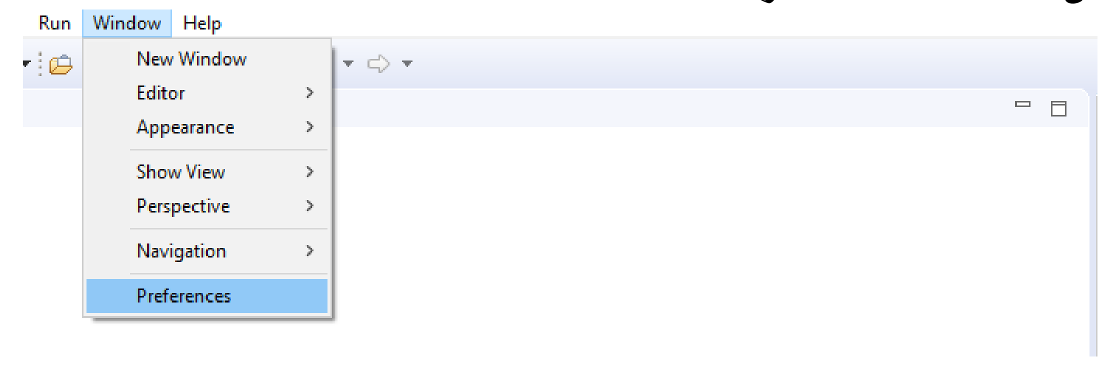

Software Updates
Would you like to restart Eclipse IDE to apply the changes?
Restart Now
No

ربط Eclipse بمحرك لغة البايثون

من الأمور المهمة ربط البرنامج بمحرك اللغة الذي قمنا بتثبيته سابقا راجع درس تثبيت بايثون يفيد الربط:

1. تشغيل ملفات Python
2. الإشارة إلى الأخطاء الخاصة باللغة أثناء كتابة الكود
3. الاكتمال التلقائي للدول بمجرد كتابة أول اخرف من الدلة ... الخ

من قائمة windows اختر Preferences



اتبع الخطوات كما هو موضح بالصورة

Preferences

type filter text

General
Ant
Code Recommenders
Gradle
Help
Install/Update
Java
Maven
Mylyn
Oomph
PyDev ❶
Builders
Debug
Editor
Interactive Console
Interpreters
IronPython Interpreter
Jython Interpreter
Python Interpreter ❷
Logging
PyUnit
Run
Scripting PyDev
Task Tags
Run/Debug
Team
Validation
XML

Python Interpreters

Python interpreters (e.g.: python.exe, pypy.exe). Double-click to rename.

Name	Location

❸ Browse for python/pypy exe
New with Pipenv
Config first in PATH
Choose from list
Remove
Up
Down

Packages | Libraries | Forced Builtins | Predefined | Environment | String Substitution Variables

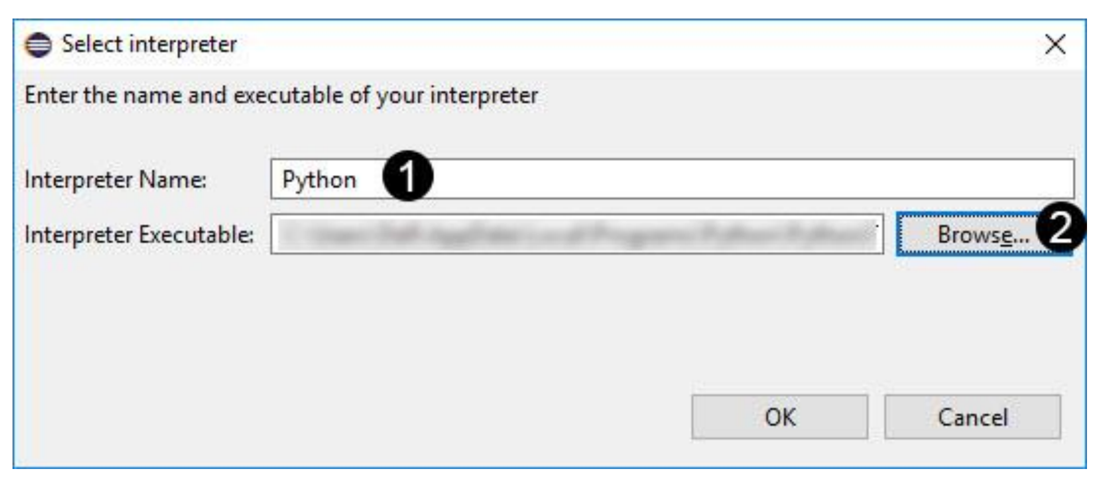
Library (pip | 9 found)	Version	
numpy	1.15.4	<pip>
pandas	0.23.4	<pip>
pip	10.0.1	<pip>
python-dateutil	2.7.5	<pip>
pytz	2018.7	<pip>
setuptools	39.0.1	<pip>
six	1.11.0	<pip>

You are using pip version 10.0.1, however version 18.1
You should consider upgrading via the 'python -m pi

Manage with pip
Manage with conda
Manage with pipenv
Load conda env vars before run?

Restore Defaults | Apply

Apply and Close | Cancel



1. قم بتسمية الإضافة
2. اختر مكان محرك لغة البايثون على جهازك تم اضغط OK

تأكد من تحديد الكل ثم اضغط OK

Selection needed

Select the folders to be added to the SYSTEM pythonpath!

IMPORTANT: The folders for your PROJECTS should NOT be added here, but in your project configuration.

Check:http://pydev.org/manual_101_interpreter.html for more details.

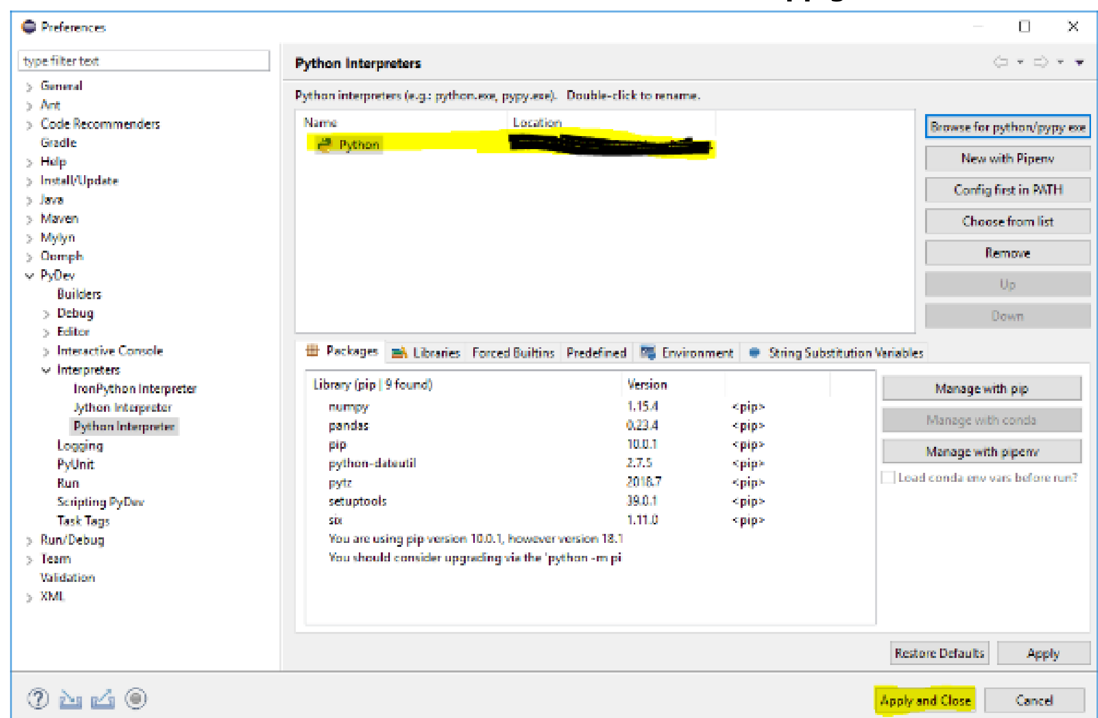
- [x] C:\Users\Dell\AppData\Local\Programs\Python\Python37-32\DLLs
- [x] C:\Users\Dell\AppData\Local\Programs\Python\Python37-32\lib
- [x] C:\Users\Dell\AppData\Local\Programs\Python\Python37-32
- [x] C:\Users\Dell\AppData\Roaming\Python\Python37\site-packages
- [x] C:\Users\Dell\AppData\Local\Programs\Python\Python37-32\lib\site-packages

Select All not in Workspace | Select All | Deselect All

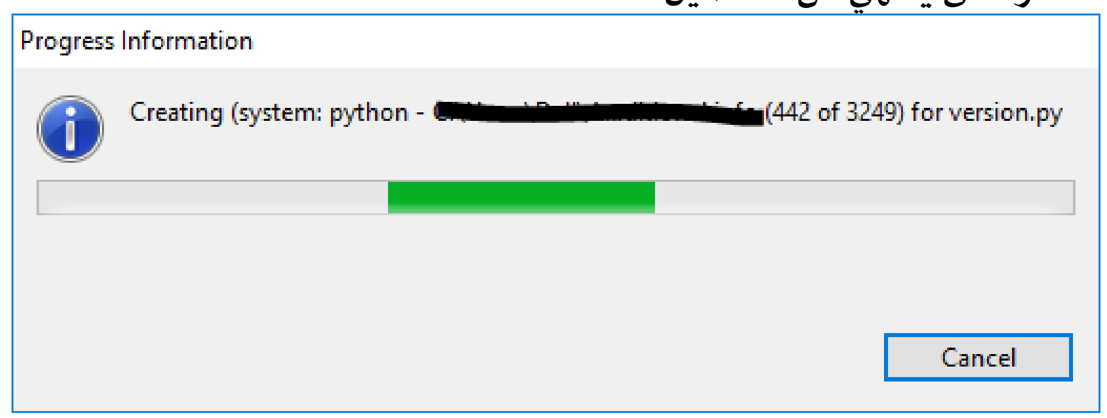
OK | Cancel

1. تأكد من تحديد كل العناصر بالضغط على زر Select All
2. اضغط على زر OK

اضغط Apply and Close



انتظر حتى ينتهي من التحميل



لعمل مشروع جديد من قائمة File -> New -> PyDev Project

PyDev Project

Create a new PyDev Project.

Project name: hello

Project contents:

☑ Use default

Directory C:\Users\Dell\eclipse-workspace\hello Browse

Project type

Choose the project type

◉ Python ○ Jython ○ IronPython

Grammar Version

Same as interpreter

Interpreter

Default -- currently: Python

Click here to configure an interpreter not listed.

Additional syntax validation: <no additional grammars selected>. ...

◉ Add project directory to the PYTHONPATH

○ Create 'src' folder and add it to the PYTHONPATH

○ Create links to existing sources (select them on the next page)

○ Don't configure PYTHONPATH (to be done manually later on)

Working sets

☐ Add project to working sets New...

Working sets: Select...

< Back Next > Finish Cancel

قم بتسمية المشروع الخاص بك ثم Finish

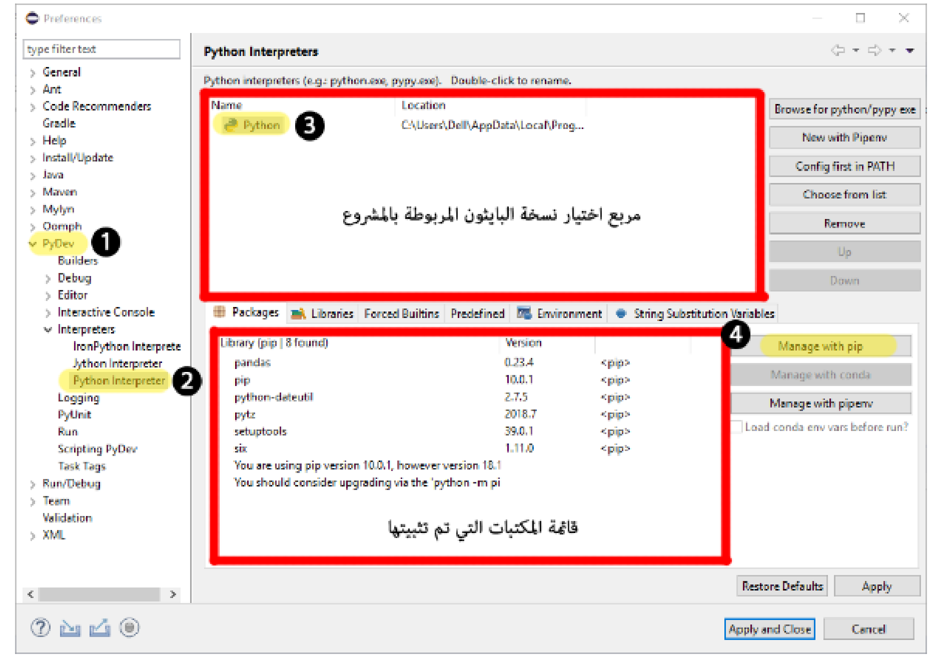
الاسم يجب أن يكون بأحرف إنجليزية وسيقوم البرنامج بإنشاء مجلد بهذا الاسم للبرنامج ويحفظه تلقائيا في مساحة عمل البرنامج المحددة مسبقا

تثبيت المكتبات باستخدام eclipse

في الدروس التالية سنحتاج إلى تثبيت بعض الإضافات Packages الملحقة بلغة البايثون فإليك الطريقة لتثبيتها عن طريق برنامج eclipse
في العادة نستخدم امر pip

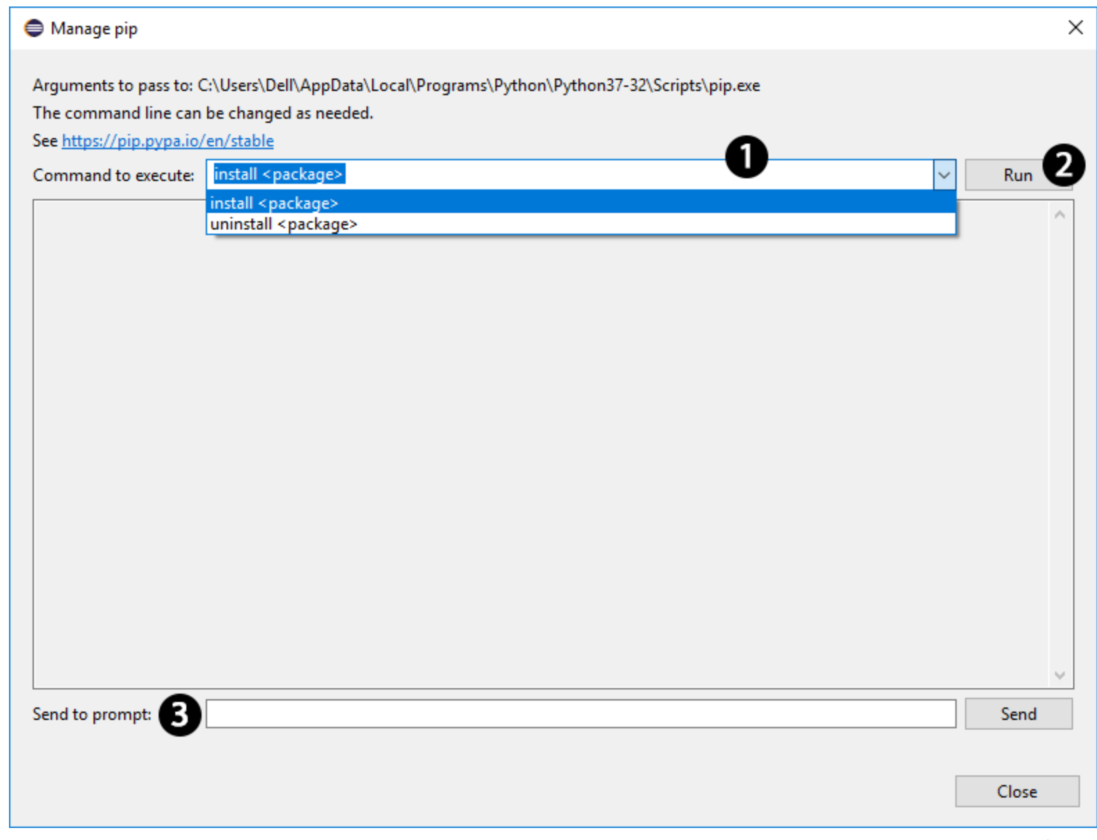
لمزيد من المعلومات حول هذا الموضوع
https://packaging.python.org/tutorials/installing-packages

سنقوم بتثبيت مكتبة numpy وهي خاصة للتعامل مع المصفوفات
من قائمة window -> References
تأكد من اتباعك الخطوات المبينة بالصورة

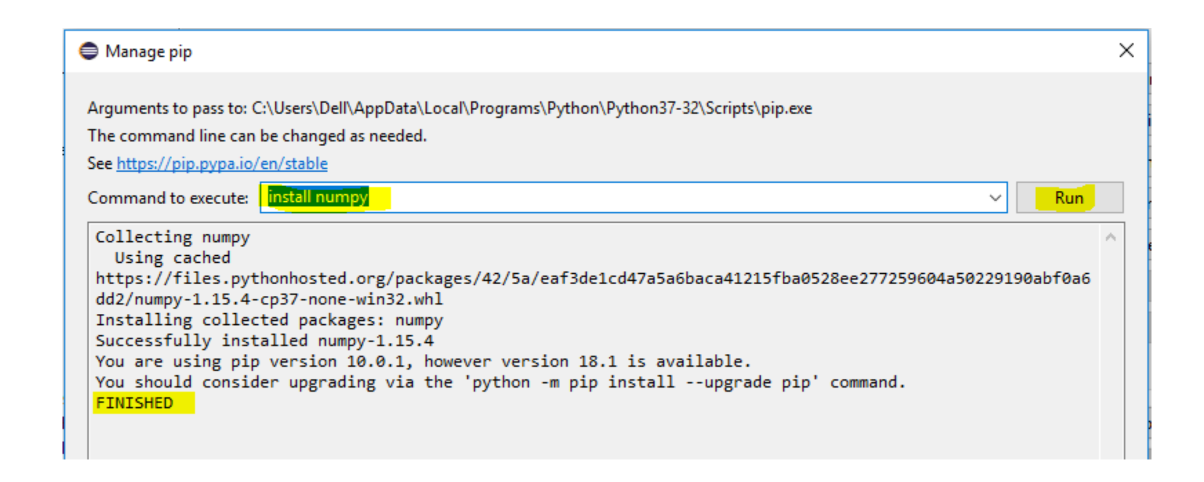


1. نختار PyDev
2. نختار interpreters <- PyDev interpreters
3. نختار محرك اللغة الذي قمنا بتركيبة مسبقاً

4. اضغط على زر Manage with pip



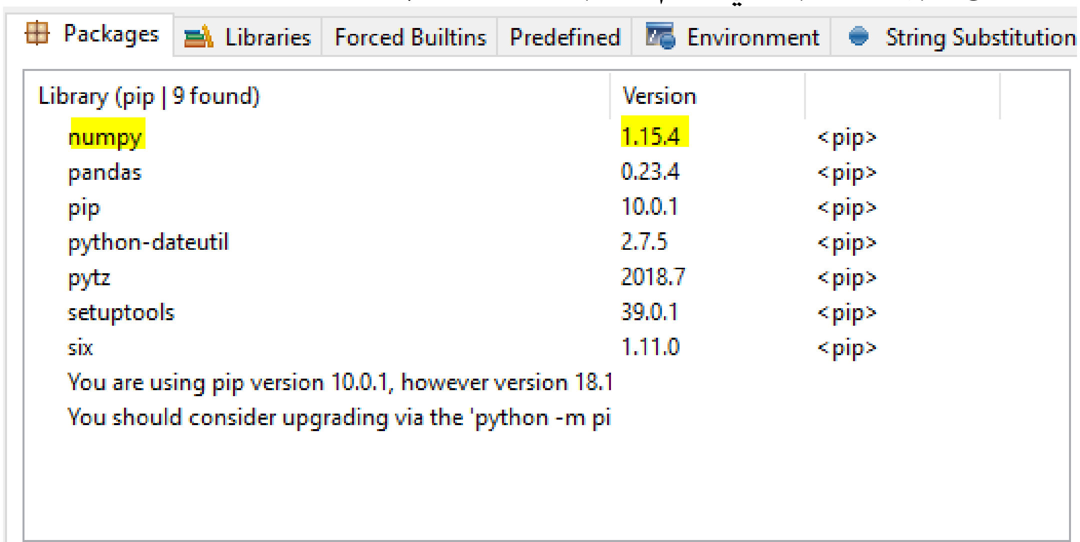
1. من القائمة المنسدلة يمكنك اختيار العملية تثبيت/الغاء التثبيت install/uninstall اختر install للتثبيت.
2. قم بتغيير <package> إلى اسم المكتبة المراد تثبيتها numpy اضغط على زر Run للتشغيل.
3. قد يتطلب من البرنامج الموافقة على بعض الإجراءات أثناء التثبيت أو الغاء التثبيت عن طريق كتابة الأمر المراد تنفيذه في مربع النص Send to prompt وتضغط Send للتنفيذ.



اضغط على زر close لأغلاق النافذة

بعض الضغط على زر run سيقوم البرنامج بتحميل ملفات المكتبة من الأنترنت ثم يقوم بتثبيتها تأكد من اتصالك بالأنترنت وسيخبرك بانتهاء التثبيت بكتابة كلمة FINISHED في أخر المربع.

تأكد من تثبيت المكتبة في قائم المكتبات الخاصة باللغة



حذف المكتبات باستخدام eclipse

في هذه المرة نختار **uninstall** من القائمة المنسدلة مع استبدال **<package>** باسم المكتبة

```
Manage pip

Arguments to pass to: C:\Users\Dell\AppData\Local\Programs\Python\Python37-32\Scripts\pip.exe
The command line can be changed as needed.
See https://pip.pypa.io/en/stable

Executing:  uninstall numpy                                         Cancel

Uninstalling numpy-1.15.4:
  Would remove:
    c:\users\dell\appdata\local\programs\python\python37-32\lib\site-packages\numpy-1.15.4.dist-info\*
    c:\users\dell\appdata\local\programs\python\python37-32\lib\site-packages\numpy\*
    c:\users\dell\appdata\local\programs\python\python37-32\scripts\f2py.py
Proceed (y/n)?

Send to prompt:  y                                                  Send

                                                                    Close
```

هنا يطلب منا تأكيد حذف الملفات الخاصة بالمكتبة نكتب **y** ثم نضغط على زر **Send**

تمت عملية الحذ بنجاح نضغط على زر **close** لأغلاق النافذة وسيقوم **eclipse** بتحديق قائمة المكتبات فور الإغلاق

```
Uninstalling numpy-1.15.4:
  Would remove:
    c:\users\dell\appdata\local\programs\python\python37-32\lib\site-packages\numpy-1.15.4.dist-info\*
    c:\users\dell\appdata\local\programs\python\python37-32\lib\site-packages\numpy\*
    c:\users\dell\appdata\local\programs\python\python37-32\scripts\f2py.py
Proceed (y/n)?

  Successfully uninstalled numpy-1.15.4
You are using pip version 10.0.1, however version 18.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
FINISHED
```

تم حذف المكتبة من القائمة بنجاح

Packages | Libraries | Forced Builtins | Predefined | Environment | String Substitution

Library (pip | 8 found)	Version	
pandas	0.23.4	<pip>
pip	10.0.1	<pip>
python-dateutil	2.7.5	<pip>
pytz	2018.7	<pip>
setuptools	39.0.1	<pip>
six	1.11.0	<pip>
You are using pip version 10.0.1, however version 18.1		
You should consider upgrading via the 'python -m pi		

لا تنسى الضغط على زر Apple And Close لتحديث المشروع بالمكتبات الجديدة

قم بتثبيت مكتبة pandas بنفسك

إنشاء ملف جديد وتشغيله

قم بفتح برنامج **eclipse** من خلال قائمة **start** ثم اختر قائمة **File->New->file**

New File

File

Create a new file resource.

Enter or select the parent folder:

hello

hello

zakat

1

File name: test.py

2

Advanced >>

3

Finish

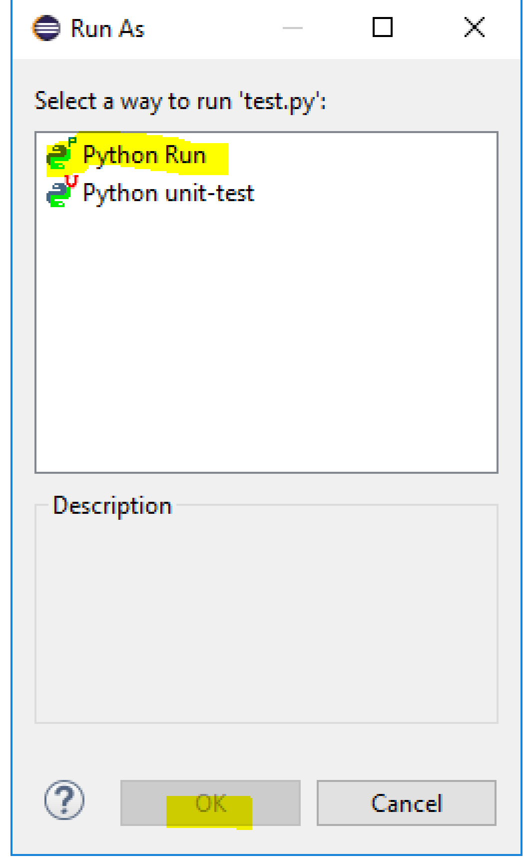
Cancel

1. قم بتحديد المشروع الذي تعمل علية
2. اكتب اسم الملف لابد أن ينتهي ب py.
3. تم اضغط Finish

اكتب الكود التالي

```
print("This is Python")
```

قم بتشغيل الملف من الزر الأخضر Run أو من قائمة Run->Run أو بالضغط على Ctrl+F11 من لوحة المفاتيح.



اختر Python Run ثم اضغط Ok

لاحظ النتيجة ظهرت في الconsole اسفل البرنامج

```
Console  PyUnit  Properties
<terminated> test.py [C:\Users\Dell\AppData\Local\P
This is Python
```

إن لم تكن موجودة اذهب إلى القائمة **Window->Show View->Console** لإظهارها

```
1 #errors
2 Undefined variable: dgjkh
3 name = "Python"
4
5
```

بعض ميزات برنامج Eclipse

تتبع وتصيد الأخطاء

لاحظ العلامة الحمراء على عمود أرقام الأسطر في حال ارتكابك خطا في برنامجك يقوم eclipse بتنبيهك بوضع هذه العلامة وإذا أوقفت عليها الفارة ستظهر لك نافذة صفراء توضح لماذا حدث الخطاء .

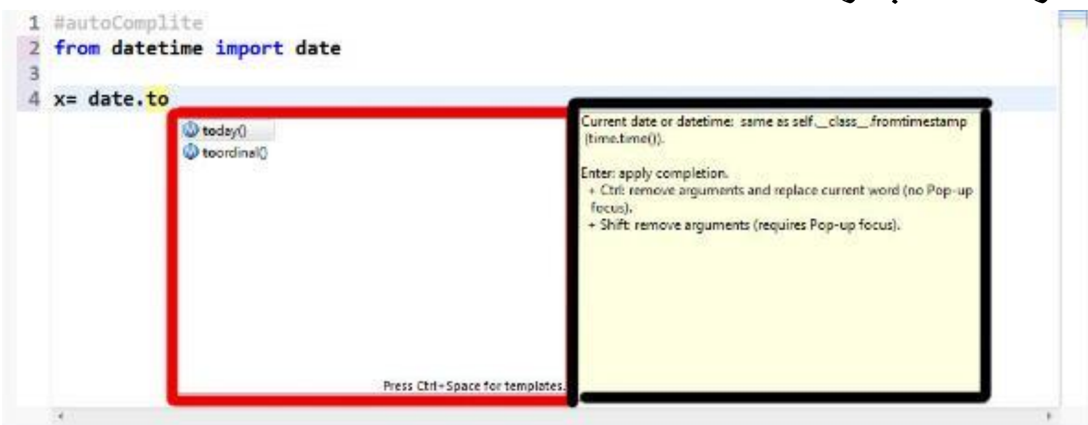
الاكتمال والتصحيح التلقائي

يقلل الجهد والوقت والأخطاء.

يقوم eclipse بتخزين مدلولات اللغة من دوال ومكتبات وموديلات ... الخ فمن خصائصه

1. إظهار قائمة منسدلة بالدوال أو المتغيرات التي تتشابها مع الحروف التي تكتب.
2. إكمال كتابة كود الشرط والتكرار والاستدعاء ... الخ

مما يساعدك كمطور في تقليل الوقت والجهد المبذول كما يساعدك على حفظ مفردات اللغة بسرعة



في المربع باللون الأحمر قائمة منسدلة بالدوال أو المتغيرات المرتبطة بحرفين **to**

> لاحظ تظليل باللون الأسمر الغامق الأحرف المتشابهة
> للتبديل بين الخيارات في القائمة المنسدلة استخدم أسهم لوحة التحكم الأعلى والأسفل

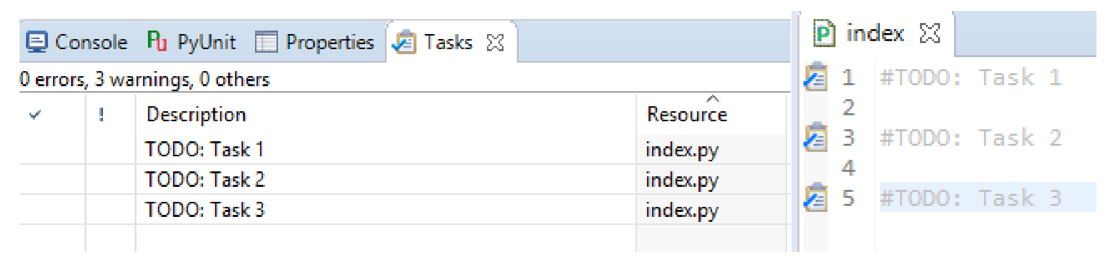
المربع الأصفر شرح لماهية العنصر المختار.

قائمة المهام Tasks LIST

من الأمور الجيدة في برنامج **eclipse** قائمة المهام **Tasks** فأثناء عملك على مشروع كبير قد توجل كتابة نقطة مهمة في الكود لحين الانتهاء من نقطة أخري ولكيلا تسقط النقطة الأولي منك فعليك كتابة كلمة :TODO بأحرف كبيرة مسبوقة بعلامة # (لماذا نكتبها بعد علامة #؟)(**الإجابة في درس التعليقات**) ملحقة بالمهمة المطلوب تنفيذها.

بمجرد حفظ الملف ستظهر علامة دفتر بجوار السطر التي كتبت فيه على شريط أرقام الأسطر لعرض القائمة من قائمة

Window->Show View->Tasksستظهر بجوار console في المربع اسفل الملف



بالضغط على المهمة سيوجهك البرنامج إلى موضوع المهمة في الملف (السطر المسجلة فيه المهمة).

للمزيـــــد بشـــــان هـــــذا الصــــدد اذهـــــب إلـــــي الــــــرابط التـــــالي http://www.pydev.org/manual_adv_features.html

المتغيرات Variables

يعتبر المتغير من أساسيات أي لغة برمجة هو عبارة عن مخزن للقيمة يقوم بالاحتفاظ بالقيمة المضافة له في الذاكرة، ويمكن الرجوع إليه واستخدامه، ويمكن تعريفه بأكثر من نوع مثل عددي أو رقمي .... الخ، كما يمكن تحديث هذه القيمة مستقبلا.

على سبيل المثال لو اعتبرنا أن الكاس (الكوب) عبارة عن متغير كونه فارغا لا يحتوي على شيء إذاً لا يحتوي على قيمة وبمجرد ملئه بالماء أو الشاي (القيمة المتغيرة) أصبح يحتوي على قيمة يمكن تقديمه إلى الضيوف أو استخدامه في شيء أخر، وبالإمكان تغيير القيمة التي يحتويها إلى العصير مثلا.

تعريف المتغير (إنشاء متغير)

اسم المتغير = القيمة

علامة تساوي (=) تفيد بان القيمة التي تليها هي القيمة المعرفة للمتغير (المخزنة في الذاكرة)

```
#display my age & my name

my_age = 10
print(my_age)
my_name = "Hassan"
print(my_name)

```

```
10
Hassan
```

دالة print تستخدم في طباعة القيمة الموجودة بين القوسين ()

تعيين قيمة جديدة لمتغير

في بعض الأحيان تحتاج إلى استدعاء المتغير في مكان أخر من البرنامج وتغيير قيمته مجرد أن تقوم بإعطاء المتغير قيمة جديدة تستبدل القيمة الجديدة في الذاكرة بدلا من القيمة القديمة

```
# display my age

my_age = 10
print(my_age)
my_age = 20
print(my_age)
```

```
10
20
```

تعريف(إنشاء) أكثر من متغير في سطر واحد

وذلك بوضع المتغيرات بين علامة فاصلة (,)

```
#define multi var with same line

my_age,my_sallery,my_degree = 10,20,30
print(my_age)
print(my_sallery)
print(my_degree)
```

```
10
20
30
```

تعريف أكثر من متغير بنفس القيمة

```python
#define multi var with same value

my_age=my_sallery=my_degree = 10
print(my_age)
print(my_sallery)
print(my_degree)

```

```
10
10
10
```

الثوابت

عبارة عن متغير يحمل قيمة لا يمكن تغييرها فيما بعد.
مثال اسم قاعدة البيانات لو تم تغييره في مرحلة من مراحل برمجة النظام بالخطأ سيتعطل الاتصال بقاعدة البيانات وتجريب البرنامج.
لذلك يجب جعل اسم قاعدة البيانات في متغير لا يمكن تغييرة حتى لو حاولت إعادة تعريفة بالخطأ.

يتم تعريف الثوابت بكتابتها بأحرف كبيرة
يفضل وضع كل الثوابت التي ستستخدمها في جهازك في مديول (ملف) خاص بها

> عمليا لا تستخدم الثوابت في بايثون (عكس اللغات الأخرى) ولكننا نحاول استخدامها بأحرف كبيرة كلية لنستبعد إعادة تعريفها

قمنا بإنشاء ملف باسم constant.py وضعنا فيه ثابت لاسم قاعدة البينات

```
#constant file
DATABASENAME = "school_library"
```

```
#call constant file
import constant as cons

print("database name id {}".format(cons.DATABASENAME))
```

```
database name id school_library
```

استدعينا ملف constant.py في الملف الثاني وقمنا بطباعة اسم قاعدة البيانات

قواعد كتابة المتغيرات والدوال والكلاسات

هناك بعض الدوال المحجوزة للغة البايثون فلا يمكنك إعادة تعريفها من جديد كدوال أو متغيرات خاصة بك

مثال: دالة **while** هي دالة محجوزة للغة البايثون ووظيفتها عمل تكرار بين نقطة انطلاق ونقطة نهاية **Loop** ولو حاولنا إعادة تعريفها كدالة خاصة بنا كما في الكود التالي سيطبع لنا خطا **invalid syntax** وهو خاص بخطأ كتابة ي اللغة

```
def while(a):
    print(a)
while(10)
```

```
sdef while(a):
              ^
SyntaxError: invalid syntax
```

من اجل تعريف (تسمية) المتغيرات والدوال والكلاسات في لغة البايثون لابد من اتباع قواعد خاصة لتعريفها

1) بحروف لاتينية فقط من a-z
2) حروف كبيرة وصغير **getAge & SetAge**
3) لا يبدا بأرقام أو رموز **1getAge** أو **@getAge**
4) يمكن استخدام الأرقام أخر الاسم وليس أوله مثال **getAge1**
5) للفصل بن كلمة وكلمة استبدل المسافة ب (__) مثال **is_pray_in_time**
6) حاول استخدام كلمات معبرة ومفهومة في اختيارك أسماء المتغيرات ليسهل التعرف عليها مثال **age=10** وليس **a=10**
7) لا يعتبر هذان المتغيرات متغير واحد **Age age** لغة البايثون حساسة لحالة الأحرف

يمكنك استخدام camel-case لتسمية المتغيرات وتكمن طريقتها في استبدال المسافات باستخدام أول حرف كبير للكلمة التالية مثال isPrayInTime
ولكن يفضل استخدام طريقة الـ (__)

للمزيد حول الموضوع يمكنك التوجه إلى الرابط التالي https://www.python.org/dev/peps/pep-0008/

التعليقات

هي عبارة عن نص مكتوب تسبقه علامة محددة لأخبار محرك اللغة أن هذا النص لا يدخل ضمن نطاق عمل الكود.
تستخدم التعليقات داخل الكود عادة لتوضيح امر معين للأخرين في حالة العمل على مشاريع ضخمة تتضمن مشاركة أكثر من شخص في المشروع .

كيفية كتابة التعليق

ا- تعليق سطر واحد

وهو نص مسبوق بعلامه #تخبر علامة #البرنامج أن السطر الذي يليها تعليق لا يدخل ضمن نطاق عمل كود البرنامج.(مستثني من البرنامج)
كما يمكنك استخدامه في أي موضع من السطر وسطه أو أوله أو أخرة

1. في حالة استخدامه في وسط السطر هذا يعني أن باقي السطر غير داخل في نطاق عمل الكود
2. Eclipse يعطي لون رصاصي لتمييز التعليق من الكود الأصلي

```python
# function return my age

def my_age(a):
    print(a)
```

2- تعليق أكثر من سطر

لكتابة تعليق أكثر من سطر يتم وضع علامة (""") أو (''') في مقدمة النص وأخرة

```python
"""=== function to return my age =====
    don't us string value to age like "12"
"""

def my_age(a):
    print(a)
```

أنواع البيانات

كل متغير في لغة البايثون يحمل قيمة وكل قيمة لها نوع فعلى سبيل المثال لو قمنا بعمل برنامج بسيط لتسجيل المواليد لاحتجنا بيانات المولود مثل الاسم والعمر واسم الأب ورقم هوية الأب .

فالاسم نوعه نصي لا يمكن أن يكون رقمي ورقم الهوية رقمي لا يمكن أن يكون نصي وفيما يلي سيتم شرح كل نوع وخصائصه وطريقة التعامل معه في لغة البايثون.

رقمي – Numbers

تعتبر لغة البايثون أن الأرقام (**integer**) والأرقام العشرية (**float**) نوعها رقمي مثال **1**و**2**و**2.02**

> ملحوظة: الأرقام لا يمكن استخراج الطول الخاص بها كما في النصي فهي محفوظة في الذاكرة بنفس القيمة فلا استخدام
> دالة len معها

نصي - Strings

يعتبر من أشهر الأنواع المستخدمة في كتابة الكود وهو عبارة عن **Unicode characters.**

يمكنك تعريفة بوضع النص المراد بين علامتين **"String here" & 'String here'**

النص عبارة عن مصفوفة (سنوفيها بالشرح في الدروس اللاحقة) حيث يعتبر كل حرف ك **index** (معرف رقمي)في هذه المصفوفة يتم استدعائه كما يتم استدعاء العناصر من الـ **list**

```python
# Declaring String.

first   = 'First String here'
second  = "Second String here"

print (first)
print (first [0])
print (first [1])

print (second)
print (second [0])
print (second [1])
```

```
First String here
F
i
Second String here
S
E
```

يمكن جمع نصين مع بعضمهما في متغير واحد باستخدام علامة الجمع +

```python
# work in string

str1 = "This lang call "
str2 = "Python"
print(str1+str2);
```

```
This lang call Python
```

لا استبدال نص: بنص أخر نستخدم دالة **replace (old, new)** حيث **old** النص المراد استبداله و **new**النص البديل.

```
# replace string

str1 = "This lang call Java"
str2 = "Python"
print(str1.replace("Java",str2));
```

```
This lang call Python
```

قائمة - List

القوائم تعتبر من اهم الأنواع في بنية البيانات وكثيرا ما ستستخدمها في البرامج الخاصة بك وسوف نفرد لها قسم خاص لشرحها باستفاضة.
تعرف القائمة بوضع القيم التي تحتويها بين قوسين [] والفصل بين كل قيمة وقيمة ب رمز فاصلة (,)

```
# Declaring List.

list = [1,20,3000,1.05,"python"]

```

القائمة من النوع المرن يمكن الإضافة عليها والتعديل والحذف منها وتغيير نوع البيانات.

بإمكانك استخدام أكثر من نوع للداتا في نفس القائمة

استدعاء عناصر الlist

يتم استدعاء عناصر الlist بهذه الطريقة

```
#call items from list
list_name[index_start:index_count,step]
```

حيث index_start هو العنصر المراد البداية منه وindex_ count عدد العناصر بعد index_start وstep عدد تغطي العناصر في هذه الطريقة يبدا الindex من0.
قد تكون قيمة index_start أو index_end بالسالب ملحقة برقم معني ذلك استدعاء العناصر من البداية أو النهاية بحذف مقدرا القيمة السالبة من الlist .

```
1    #call items from list
2    list = [1,20,3000,1.05,"python"]
3
4    print(list[0:])
5    print(list[0:3])
6    print(list[2:3])
7    print(list[:3])
8    print(list[3:])
9    print(list[0:-1])
10   print(list[-1:])
11   print(list[2:-1])
12   print(list[0:5:2])
```

```
[1, 20, 3000, 1.05, 'python']
[1, 20, 3000]
[3000]
[1, 20, 3000]
[1.05, 'python']
[1, 20, 3000, 1.05]
['python']
[3000, 1.05]
[1, 3000, 'python']
```

- السطر 4 استدعينا العناصر من البداية إلي النهاية
- السطر 5 استدعينا من العنصر صفر إلي العنصر رقم 3
- السطر 6 استدعينا من العنصر 2 إلي 3
- السطر 7 استدعينا من البداية إلي العنصر رقم 3
- السطر 8 استدعينا من العنصر رقم 3 إلى أخر عنصر
- السطر 9 استدعينا كل العناصر بحذف أخر عنصر
- السطر 10 استدعينا أخر عنصر
- السطر 11 استدعينا من العنصر 2 إلى أخر الـlist مع حذف أخر عنصر
- السطر 12 استدعينا كل العناصر بتخطي عنصر في كل مرة

الـlist عند استدعاء العناصر بهذه الطريقة يبدا لترقيم من 0 بمعني أن 3 هنا تساوي 1.05 وليس 3000

إضافة عنصر الlist

دالةappend تستخدم لإضافة عنصر جديد إلى list

```
#add item to list
list = [1,20,3000,1.05,"python"]
list.append("Java")
print(list)
```

```
[1, 20, 3000, 1.05, 'python', 'Java']
```

لو أردت إدراج عنصر جديد في موضع معين نستخدم دالة insert(Index, value) حيث indexموضع العنصر وvalue هي قيمة العنصر

```
list = [1,20,3000,1.05,"python"]
list.insert(2,100)
print(list)
```

```
[1, 20, 100, 3000, 1.05, 'python']
```

لحذف عنصر من ال list نستخدم دالة remove أو del الأولي تعمل مع قيمة العنصر والثانية مع index العنصر

```
list = [1,20,3000,1.05,"python"]
list.remove("python")
del(list[0])
print(list)
```

```
[20, 3000, 1.05]
```

لتكرار list نقوم بضربها في عدد مرات التكرار

```
list = [1,20,3000,1.05,"python"]
print(list*2)
```

```
[1, 20, 3000, 1.05, 'python', 1, 20, 3000, 1.05, 'python']
```

مصفوفة – Tuple

تشبه إلى حد كبير القوائم (list) في خصائصها إلا أنها لا يمكنك التعديل عليها سواء بالإضافة أو الحذف أو التعديل على القيم بداخلها.

- يتم تعريفها باستخدام قوسين ()

```
# Declaring Tuple.

tuple = (1,20,3000,1.05,"python")

```

Set

تشبه كثيرا Tupleولكنها تحمل قيم متفردة غير متكررة

- يتم تعرفيها باستخدام {} والفصل بين القيم ب (,)
- تقوم بإرجاع القيم بترتيب عشوائي في كل مرة تقوم بطباعتها

```
# Declaring Set

set = {10,10,20,20,30,30,"python","python"}
print (set)

```

```
{10, 20, 30, 'python'}
في المرة التالية لطباعتها
{'python', 10, 20, 30}
```

قاموس - Dictionary

يستخدم بشكل عام عندما يكون لدينا كمية هائلة من البيانات.

- يتم تعريفة بوضع القيم داخل علامتين {}.
- كل قيمة يتم تعريفها ب key: value
- القيمة يمكن أن تحمل أي نوع من البيانات
- يتم استدعاء القيمة بمعلومية المفتاح key

```
# Declaring Dictionary
Dictionary = {"age":10,"name":"hassan","salary":100.20,10:"discount",10.2:"float key"}
print (Dictionary);
print (Dictionary ["age"])
print (Dictionary [10]);
print (Dictionary ["name"])
print (Dictionary [10.2])
```

```
{'age': 10, 'name': 'hassan', 'salary': 100.2, 10: 'discount', 10.2: 'float key'}
10
discount
hassan
float key
```

بعض دوال التعامل مع القواميس		
الدالة	الشرح	
copy	اخد نسخة من القاموس	
items	جلب كل عناصر القاموس	
get(key, default)	جلب عنصر بمعلومية الكي الخاص به	**Key** هو معرف العنصر و**default** قيمة افتراضية في حالة عدم وجود العنصر
clear	حذف كل عناصر القاموس	

keys	جلب كل مفاتيح القاموس	dict_keys(['age', 'name', 'salary', 10, 10.2])
values	جلب كل قيم القاموس	dict_values([10, 'hassan', 100.2, 'discount', 'float key']).
update	تحديث القاموس	
pop	حذف عنصر من القاموس بمعلومية الـkey واخذ نسخة منه	

تداخل القواميس

يمكن لكل مفتاح أو key في القاموس (Dictionary)أن يحمل قاموس (Dictionary) كامل ويتم استدعاء القيم بكتابة المفتاح key للقاموس الأساسي وkey القاموس الفرعي

```
#Dictionary

dictionary =
{"names":{"one":"Sayed","tow":"Ali","three":"Hassan"},
              "degree":{"Sayed":100,"Ali":80," Hassan":20}}

print(dictionary)

name = dictionary["names"]["one"]
degree = dictionary["degree"]["Sayed"]
another_degree = dictionary["degree"][name]

print("one name is {} his degree is {} type another way
{}".format(name,degree,another_degree))

```

```
{'names': {'one': 'Sayed', 'tow': 'Ali', 'three': 'Hassan'}, 'degree':
{'Sayed': 100, 'Ali': 80, 'Hassan': 20}}
one name is Sayed his degree is 100 type another way 100
```

في السطر 10 قمنا باستدعاء درجة الطالب "Sayed" من خلال متغير name المعرف في سطر 8
بما أن متغير name قيمته تساوي نفس قيمة مفتاح key الدرجات في degree وهو Sayed فبالتالي عرض القيمة 100
ارجوا أن تكون قد فهمت تداخل القواميس تتشابه هذه النقطة مع المصفوفة راجع درس المصفوفات

رموز العمليات الحسابية

الرمز	الاسم	الوصف	مثال	النتيجة
=	يساوي	تستخدم لإعطاء قيمة للمتغير	name = "Python" print(name)	Python
+	للجمع	لجمع عددين أو نصين	print(5+5) print("I Learning "+"Python")	10 I learning Python
-	للطرح	تستخدم مع الأرقام فقط	print(10-5)	5
*	للظرب	تستخدم لضرب الأعداد أو تكرار المتغيرات بعدد معين من المرات	print(10*5) print("Python "*2)	50 Python Python
/	للقسمة	تستخدم في قسمة الأعداد	print(10/5)	2.0
%	إيجاد المعامل	للتأكد ما اذا كان الرقم الأيسر يقبل القسمة على الرقم الأيمن	print(10/5)	2.0
//	القسمة	إيجاد الرقم الصحيح لناتج القسمة	print(52//10)	5
**	الأس	إيجاد الأس التربيعي .....الخ	print(2**2)	4
العمليات التالية لا تختلف كثيرا عن العمليات السابقة				

+=	زيادة	زيادة القيمة بعد = إلي قيمة العنصر	num = 10; num+=10 print(num)	20
-=	طرح	طرح القيمة بعد = من قيمة العنصر	num = 10; num-=10 print(num)	0
*=	ضرب	ضرب القيمة بعد = في قيمة العنصر	num = 10; num*=10 print(num)	100
/=	قسمة	قسمة القيمة بعد = على قيمة العنصر	num = 10; num/=10 print(num)	1.0
%=	تقبل القسمة		num = 10; num%=10 print(num)	0
//=	القسمة		num = 10; num//=10 print(num)	1
**=	الأس		num = 10; num**=10 print(num)	10000000000

التحويل بين أنواع البيانات

أثناء عملك على مشروع ضخم تحتاج في بعض الأوقات إلي التغيير بين أنواع المتغيرات بتحويل النصي إلي رقمي أو الرقمي إلي نصي.... الخ.

بايثون تقوم بذللك باستخدام دوال

- int() للتحويل إلي number
- float() للتحويل إلي float
- str() للتحويل إلي string
- dict() للتحويل إلي Dictionary
- tuple() للتحويل إلي tuple
- list() للتحويل الـ list

دوال التحويل في الغالب تسمي باسم نوع الداتا.

```
# Conversion between data types

_float = 10.20
_int   = 10
_string = "10"
_list = [1,2,3]

#float to int
float_to_int = int(_float)
print(float_to_int)
print("float to int type is ",type(float_to_int))

#int to float
int_to_float = float(_int)
print(int_to_float)
print("int to float type is ",type(int_to_float))

#string to int
string_to_int = int(_string)
print(string_to_int)
print("string to int type is ",type(string_to_int))

#int to string
int_to_string = str(_int)
print(int_to_string)
print("int to string type is ",type(int_to_string))

#list to set
list_to_set = set(_list)
print(list_to_set);
print("list to set result is ",list_to_set)

#list to set
list_to_tuple = tuple(_list)
print(list_to_tuple);
print("list to tuple result is ",type(list_to_tuple))


#list to set
```

```
list_to_dict = dict([[1,2],[3,4]])
print(list_to_dict);
print("list to Dictionary  result is ",type(list_to_dict))

#string to list
string_to_list = list("python")
print(string_to_list);
print("string to list  result is ",type(string_to_list))
```

```
10
float to int type is <class 'int'>
10.0
int to float type is <class 'float'>
10
string to int type is <class 'int'>
10
int to string type is <class 'str'>
{1, 2, 3}
list to set result is {1, 2, 3}
(1, 2, 3)
list to tuple result is <class 'tuple'>
{1: 2, 3: 4}
list to Dictionary result is <class 'dict'>
['p', 'y', 't', 'h', 'o', 'n']
string to list result is <class 'list'>
```

دالة – type

تستخدم الدالة لإرجاع نوع المتغير str,int ...etc.

```
# test using type function

my_name = "Hassan"
my_age  = 32

print ("My Name Data Type is",type(my_name))
print ("My Age  Data Type is",type(my_age))

```

```
My Name Data Type is <class 'str'>
My Age Data Type is <class 'int'>
```

الاستدعاء - import

تعتمد لغة البايثون على كثير من المديولاتmodules التي تقوم بوظائف معينة مما تساعد مطوري برامج بايثون على تطوير برامجهم بكل سهوله.

- يتكون المديول من ملف.
- للقيام بعملية الاستدعاء نستخدم كلمةimport ملحقة باسم المديول.
- عمليات الاستدعاء دائما ما تكون في اعلى الصفحة لابد من استدعاء المديول اعلى الكود المستخدم فيه.

مثال مديول datetime للتعامل مع التاريخ حيث يمكنك استداء تاريخ اليوم أو الوقت أو طباعة السنة الحالية

```
# call module datetime

import datetime

# output today
print(datetime.datetime.today())

```

```
2018-11-26 14:24:47.516117
2018
11
```

```
# output month
_now = datetime.datetime.now();
print(_now.month)
```

تنويه: في الدروس القادمة سنتعلم كيف ننشئ مديول خاص بنا

المكتبات – Packages

تعتبر المكتبات من افضل الأدوات لمطوري لغة البايثون فهي مجموعة من الملفات داخلها دوال أو كلاسات تؤدي وظائف معينة سنتعرف اكثر على المكتبات في درس التعامل مع التاريخ datetime للمزيد حول المكتبات https://pypi.org

استدعاء المكتبة داخل متغير

بعض المكتبات تكون مسماة بأسماء طويلة (عدد حروف طويلة) فيمكنك اختصار الاسم الطويل عن طريق كلمة (as) تليها الاسم المراد إعادة تسمية المكتبة به ويستخدم هذا الاسم كمتغير في الكود أي أن تقول للبرنامج استعدي هذا لمكتبة داخل هذا المتغير

```
import numpy as np
```

أصبح الأن متغير np بديل عن كتابه numpy ويحمل كل خصائص المكتبة

استدعاء ملف واحد من مكتبة

قد ترغب في استخدام ملف واحد من المكتبة فلا يتطلب الأمر استدعاء المكتبة كاملة. لو شبهنا المكتبة كمكتبة كتب والبرمجة قسم فيها وذهبت الهيا لتستعير كتاب تعليم لغة البايثون فمن البديهي استعارة كتاب البايثون فقط وليس كامل قسم البرمجة.

- تتم طريقة الاستدعاء بكتابة امر الاستدعاء from ملحق باسم المديول ملحق بكلمة import ثم اسم الملف المطلوب
- from module _name import file_name

تلاحظ في المثال السابق كلمة datetime مكرر مرتين مفصول بين كل كلمة بنقطة. هكذا تقول للبرنامج قم باستدعاء مديول datetimeوالملف داخل المديول المسمى datetime

```python
# call module datetime

from datetime import datetime

# output today
print(datetime.today())

# output day
_now = datetime.now();
print(_now.year)

# output month
_now = datetime.now();
print(_now.month)

```

```
2018-11-26 14:35:27.581727
2018
11
```

لاحظ بعد الغاءdatetime المكررة نفس النتيجة ذلك لننا قلنا استدعى datetime فقط الموجودة في مديول datetime

قم باستدعاء date الموجود في مديول datetime

```python
# call module datetime

from datetime import date

# output today
print(date.today())

```

```
2018-11-26
```

الإدخال والإخراج - input/output

دالة print للإخراج

- تستخدم دالة print لطباعة الكود الخاص بك أو إخراج نتيجة عمل برنامجك
- كثيرا ما سنستخدم دالة print لكتابة مستخرج البرنامج كالاسم أو السن .... الخ
- يمكن استخدام أي أنواع من الداتا ك numbers,strings,list,... etc.

تكتب هكذا: -

```
# using print

print(*objects, sep=' ', end='\n', file=sys.stdout,
flush=False)
```

1. sep: العلامة الفاصلة بين كل كلمة وأخري
2. end: ما سيتم عرضة أخر النص
3. file: ملف الكتابة
4. flush: ماذا كانت الداتا ستخزن أو لا الافتراضي غير مخزنة

إخراج أكثر من نوع بيانات في سطر واحد

من خلال استخدام علامة فاصلة (,)

> ملحوظة: ما بعد السطر قم 3 , يتم طباعته بشكل متتالي في أخر الجملة الأولي

```
# using print

print("my age is",10,sep="---",end=" >>>\n")
print(10," is my age")
print("my age is and my favorite Programing lang is",10,"Python")
```

```
my age is---10 >>>
10 is my age
my age is and my favorite Programing lang is 10 Python
```

تلاحظ أننا في السطر رقم **3** لم نطبع جملة ليست ذات معني لتحديد ترتيب العنصر المراد طباعتها في الجملة نستخدم علامة {} واستخدام دالة **format**

```
# using print

print("my age is {} and my favorite Programing lang is {}".format(10,"Python"))
```

```
my age is 10 and my favorite Programing lang is Python
```

- لاحظ استخدام دالة **format** بعد المتغير مباشرة مدمجة مع المتغير باستخدام نقطة (.)
- استبدلت الدالة أول قوسين بأول متغير فيها والثاني بالثاني وهكذا

لو أردنا تحديد مكان المتغيرات لاحظ الأمثلة التالية بعناية

- ❖ استخدام **%s , %d** حيث **d** للرقم و **s** للنص.

```
# using print

print("My age is %d and my favorite Programing lang is %s" %
(10, "python"))
 استخدام%()s  ال  مع dictionary
print("My age is %(age)s and my favorite Programing lang is
%(lang)s " % {'age': 10, 'lang': "python"})
 استخدام{index}  حيثindex  دالة في العنصر ترتيب هوformat
print("My age is {0} and my favorite Programing lang is
{1}".format(10, "python"))
 استخدام{var}  حيثvar   دالة في تعريفها يتم format
print("My age is {age} and my favorite Programing lang is
{lang}".format(age=10, lang="python"))
 استخدامstr  حيث نفسه النص داخلstr نصي الي متغير أي تحول
print("My age is " + str(10) + " and my favorite Programing lang
is " + str("python"))
،استخدام
print("My age is", 10, "and my favorite Programing lang is",
"python")

```

```
My age is 10 and my favorite Programing lang is python
My age is 10 and my favorite Programing lang is python
My age is 10 and my favorite Programing lang is python
My age is 10 and my favorite Programing lang is python
My age is 10 and my favorite Programing lang is python
My age is 10 and my favorite Programing lang is python
```

دالة – input للإدخال

تستخدم دالة **input** لجلب بيانات من المستخدم وتخزينها في متغير ويعتبر الإدخال الجزء الذي يعطي لبرنامجك الحياة بتفاعله مع المستخدم.
في مثالنا التالي سوف نسأل المستخدم عن اسمه ونتأكد انه ادخل اسمه بصورة صحيحة ثم نقوم بطباعة الرسالة الترحيبية **hello username**

```
# using input

username = input("what is your name");
if username == "":
    username = input("please write what is your name");
print("Hello {}".format(username))

```

```
what is your name
please write what is your name Python
Hello Python
```

الشرط

يعتبر استخدام الشروط من اهم ما يستخدم في كتابة الكود البرمجي فلا يوجد برنامج لا يستخدم الشرط
وفي تعاملاتك اليومية تستخدم الشرط
مثال: إذا كانت الفاكهة غالية فلن ابتاع اليوم
تستخدم لغة بايثون if & else and elifكادوال للشرط.

If

وهي تستخدم لتحقق من تطابق الشرط بمعني لو تحقق (الشرط) قم بتنفيذ (الأمر)
يتم تعريفها

```
if(الشرط):
    #code here
```

أو

```
if الشرط :
      #code here
```

- **لابد أن يكون الشرط بين if وعلامة (:)**
- **سطر تنفيذ الأمر يجب أن يبعد مسافة بادئة من اليسار حتى تتعرف لغة بايثون أن هذا مجال تنفيذ الشرط (نقطة مهمة جدا)**

في هذا المثال سنقوم بعمل برنامج بسيط يذكرك بصلاة الفجر إذا دخل وقتها.

```
# if conditions

fajr_time = "5 AM";
time_now = "5 AM"

if(time_now == fajr_time):
    print("fajr time")
```

```
fajr time
```

جدول الرموز الشرطية

الرمز	الاسم	الوصف	مثال
==	يساوي	تستخدم للمقارنة بين قيمين	If name == "Python" :
===	يساوي ومن نفس النوع	للمقارنة بين قيمتين من نفس النوع	If name === "Python" :
!=	لا تساوي	للمقارنة اذا كانت القيمة لا تساوي القمة الثانية	If name != "Python" :
not	لا تساوي	نفس != ولكن تستخدم في حالة المقارنة ب **True & False**	If name not True :
>=	اكبر من أو تساوي	للمقارنة اذا كانت قيمة اكبر من الأخرى أو تساويها	If age >= 50 :
<=	اقل من أو تساوي	للمقارنة اذا كانت قيمة اقل من الأخرى أو تساويها	If age <= 50 :
>	اكبر من	للمقارنة اذا كانت قيمة اكبر من الأخرى	If age > 50 :
<	اقل من	للمقارنة اذا كانت قيمة اقل من الأخرى	If age < 50 :
in	موجود ضمن	للتأكد أن القيمة ضمن القيمة الأخرى تستخدم غالبا مع القوائم **10 in [10,20,30,40]**	If age in [10,20,30,50]:

not in	غير موجودة ضمن	للتأكد اذا كانت القيمة غير موجودة ضمن القيمة الأخرى	`If age not in [10,20,30,50]:`
is	يساوي	إرجاع **False/True**	`var1 is var2`
is not	لا يساوي	إرجاع **False/True**	`var1 is not var2`

قم بتجربة استخدام الشروط السابقة.

elif

إذا لم يتحقق الشرط الأول تأكد من تحقق الشرط التالي ز

- **elif** تشبه في كتابتها **if**.
- لابد أن يكون الشرط بين **elif** وعلامة (:).
- سطر تنفيذ الكود هو السطر التالي لكلمة **elif** : يجب ترك مسافة بادئه من اليسار قبل سطر التنفيذ.
- دائما ما تستخدم **elif** بعد **if**للتأكد من تحقق شرط أخر.

```
# elif conditions

fajr_time = "6 AM";
time_now = "5 AM"
is_prayer_at_night = False

if(time_now == fajr_time):
    print("fajr time")
elif is_prayer_at_night == False:
print("Prayer at night")
```

```
Prayer at night
```

> في مثالنا السابق إذا كان الوقت الأن يساوي وقت صلاة الفجر صلي الفجر إذا لم يكن تأكد هل صليت قيام الليل اذا لم يكن صلي قيام الليل

else

تستخدم دائما بعد if أو elif كبديل في حالة لم يتحقق شرط if ففي مثالنا السابق إن لم يدخل وقت صلاة الفجر وقمت الليل اذا أبداء في قراءة ورد القران.

- **else تشبه في كتابتها if غير أنها لا تحتوي على شرط**
- **تكتب هكذا :else بعلامة (:) أخر كلمة else**
- **سطر تنفيذ الكود هو السطر التالي لكلمة :else يجب ترك مسافة بادئه من اليسار قبل سطر التنفيذ**
- **دائما ما تستخدم else في أخر الجملة الشرطية (كأمر أخير في حالة لم يتحقق أي شرط)**

```
# if conditions else

fajr_time = "6 AM";
time_now = "5 AM"
is_prayer_at_night = False

if(time_now == fajr_time):
    print("fajr time")
elif is_prayer_at_night == False:
    print("Prayer at night")
else:
    print("Read Quran")

```

```
Read Quran
```

في المثال إذا لم يدخل وقت الفجر وأقمت صلاة الليل أقراء القران

استخدام if داخل if

يمكنك استدعاء شرط ثاني في حالة تنفيذ الشرط الأول
في مثالنا التالي

- **إذا دخل وقت صلاة الفجر**
  - **وقمت بأداء الصلاة**
  - **إبداء بالأذكار**

وإذا لم يدخل الوقت اقرأ القران

```python
# conditions under conditions

fajr_time = "6 AM";
time_now = "6 AM"
is_prayer = True

if(time_now == fajr_time):
    if is_prayer:
        print("Start Azkar")
    else:
        print("Do Prayer")
else:
    print("read Quran")

```

```
Start Azkar
```

حاول تغيير قيمة is_prayer إلي False وانظر النتيجة

```python
# conditions under conditions

fajr_time = "6 AM";
time_now = "6 AM"
is_prayer = False

if(time_now == fajr_time):
    if is_prayer:
        print("Start Azkar")
    else:
        print("Do Prayer")
else:
    print("read Quran")
```

```
Do Prayer
```

استخدام أكثر من شرط

and

يمكنك التأكد من تحقيق شرط أو أكثر من خلال استخدام كلمة **and** مسبوقة بالشرط الأول وملحقة بالشرط التالي

```
if condition and condition and condition and ....:
```

مثال إذا كانت اللغة المفضلة بايثون والسن **20** تمت عملية البحث بنجاح".

or

كما يمكنك المفاضلة بين أكثر من شرط باستخدام **or** يسبقها الشرط الأول ويليها الشرط الثاني وهكذا.

```
if condition or condition or condition or ....:
```

مثال إذا كانت اللغة المفضلة بايثون أو السن **20** اطبع "تمت عملية البحث بنجاح".

الرمز	الاسم	الوصف	مثال
and	و	نفذ الشرط في حالة تحقق الشرطين معا في مثالنا التالي نفذ الأمر اذا كانت اللغة المفضلة = **python** والعمر = **20** سنة	If p_lang == "Python" and age == 20 :
or	أو	نفذ الشرط في حال تحقق أي من الشرطين في مثالنا التالي نفذ الأمر اذا كانت اللغة المفضلة = **python** أو العمر = **20** سنة	If p_lang === "Python" or age == 20:

```
# using and and or

fajr_time = "6 AM";
time_now = "6 AM"
is_prayer = True

if(time_now == fajr_time and is_prayer):
    print("Read Quran")
print("==============")

is_prayer = False

if(time_now == fajr_time or is_prayer):
    print("Read Quran")

```

```
Read Quran
==============
Read Quran
```

في مثالنا السابق قلنا إذا دخل وقت الفجر وقمت بأداة الصلاة إذا اشرع في قراءة القران ثم أعطينا قيمة جديدة لمتغير is_prayer وهي False وهذا المتغير من يحدد هل قمت بأداء صلاة الفجر أم لا.

استخدام شرطين في نفس السطر

إذا أردت المقارنة بين شرطين بحيث يكون احتمال تحقق أحدهما يمكنك تجميعها بين قوسين ()

```
if condition and (condition or condition) and ....:
if condition or (condition and condition) or ....:
```

يمكنك التغيير بين or وand بما يطلبه برنامجك

```
# using and and or with more condition

fajr_time = "6 AM";
time_now = "6 AM"
is_prayer = True
is_read  = True
if(time_now == fajr_time and (is_prayer or is_read)):
    print("got to your Work")

```

```
got to your Work
```

التكرار- Loop

عادة ما يستخدم لتكرار امر محدد لعدد مرات محددة من مسبقا.
احرص على عدم استدعاء التكرار بشكل لانهائي يتعطل البرنامج لاستهلاكه مساحة كبيرة من الذاكرة.

for

```
1    for var in var2:
2        #code here
```

معنى الكود السابق السطر 1 تحقق ماذا كان المتغير var يدخل ضمن نطاق المتغير var2

مثال 1- اطبع جملة "لغتي المفضلة بايثون" 10مرات

```
# using for

for i in range(0,10):
    print("My favorite lang is Python ",i)

```

```
My favorite lang is Python 0
My favorite lang is Python 1
My favorite lang is Python 2
My favorite lang is Python 3
My favorite lang is Python 4
My favorite lang is Python 5
My favorite lang is Python 6
My favorite lang is Python 7
My favorite lang is Python 8
My favorite lang is Python 9
```

دالة range تستخدم لطباعة list (قائمة) بالأرقام لنطاق محدد range
(start number, end number)

مثال 2-

- يمكنك استخدام list في التكرار سيقوم البرنامج بطباعة الجملة
- "My favorite in list" 5 مرات بعدد العناصر الموجودة الـ list.
- المتغير x قيمة نطاق التكرار.

```
# using for

lang_list = ["php","java","javascript","python","c++"]
for x in lang_list :
    print("lang in this loop line is",x)

```

```
lang in this loop line is php
lang in this loop line is java
lang in this loop line is javascript
lang in this loop line is python
lang in this loop line is c++
```

مثال 3- سنقوم بطباعة لغتي المفضلة بايثون إذا كانت اللغة موجودة ضمن قائمة اللغات وطباعة ليست لغتي المفضلة في حالة أن اللغة داخل التكرار لا تساوي لغتي المفضلة

```
# using for

l = "python"
lang_list = ["php","java","javascript","python","c++"]
for x in lang_list :
    if l == x:
        print("{} i already learning".format(x))
    else:
        print("{} not Learn yet".format(x))

```

```
php not Learn yet
java not Learn yet
javascript not Learn yet
python i already learning
c++ not Learn yet
```

استخدام التوقف (break)

تستخدم لإيقاف التكرار إذا ما تحقق الشرط المحدد او إيقاف التكرار بشكل كامل اذا ما وضعت في بداية التكرار

```
# using break

lang_list = ["php","java","javascript","python","c++"]
for x in lang_list:
    if x =="javascript":
        break;
    print(x)
```

```
php
java
```

اشترطنا في المثال السابق أن لو x تساوي javascriptأوقف اللوب فكانت النتيجة طباعة العنصرين السابقين لـ javascript وهما php, java

ملحوظة: break يختلف استخدمها في حال كانت قبل أو بعد السطر التنفيذي كما في مثالنا السابق سيتم التأكد من تحقق الشرط وهو أن اللغة هي الجافا سكريت فاذا تحقق الشرط أوقف عملية التكرار بالتالي لن يكمل عملية طباعة أسماء اللغات لان إعطاء امر التوقف قبل عملية الطباعة.

```
# using break

lang_list = ["php","java","javascript","python","c++"]
for x in lang_list:
    print(x)
    if x =="javascript":
        break;
```

```
php
java
javascript
```

في هذا المثال قدمنا دالة الطباعة على شرط توقف التكرار لذلك طبع javascript

استخدام التخطي (continue)

تستخدم لتخطي التكرار في نقطة معينة والمواصلة إلى النقطة التالية

ملحوظة: تستخدم continue قبل سطر تنفيذ الكود البرمجي ولا يمكن استخدامها بعد سطر التنفيذ البرمجي

```python
# using continue

lang_list = ["php","java","javascript","python","c++"]
for x in lang_list :

    if x =="javascript":
        continue
    print(x)

```

```
php
java
javascript
python
c++
```

```python
# using continue after code

lang_list = ["php","java","javascript","python","c++"]
for x in lang_list :

    print(x)
    if x =="javascript":
        continue

```

```
php
java
python
c++
```

لاحظ تخطى طباعة **javascript**

لاحظ تم طباعة كل لغات البرمجة الموجودة في list لاستخدمنا continue بعد سطر تنفيذ الكود البرمجي (طباعة عناصر الـ list)

While

يتواصل التكرار مادام الشرط صحيح.

```
1  # using while
2
3  loop = 0;
4  lang_list = ["php","java","javascript","python","c++"]
5  lang_length = len(lang_list);
6  print("{} is the list count".format(lang_length))
7  while loop < lang_length:
8      print(lang_list[loop])
9      loop+=1
10
```

```
5 is the list count
php
java
javascript
python
c++
```

في مثالنا السابق

1. في السطر رقم **5** قمنا بإيجاد عدد العناصر الموجودة داخل الـ **list**
2. في السطر **6** قمنا بطباعة عدد العناصر الموجودة الـ **list**
3. في السطر **7** بدانا في كتابة التكرار وهنا اشترطنا أن يكون المتغير **loop** صاحب القيمة البادئة صفر دائما اقل من مجموع عدد العناصر الموجودة في الـ **list**
4. في السطر **8** قمنا بطباعة عنصر الـ **list** بمعلومية الـ **index** بمعلومية **loop**
5. في السطر **9** قمنا بتحديث المتغير **loop** بزيادتة قيمة **1** في كل مرة من التكرار
6. دائما ما يتم زيادة قيمة متغير التكرار (في مثالنا السابق **loop**) في أخر دالة التكرار

```python

while loop < 5:
    #step -1- loop = (loop+1) = (0+1) = 1
    #step -2- loop = (loop+1) = (1+1) = 2
    #step -3- loop = (loop+1) = (2+1) = 3
    #step -4- loop = (loop+1) = (3+1) = 4
    #step -5- loop = (loop+1) = (4+1) = 5
#in step 6 loop will stop because loop > 5 condition is false
    loop+=1

```

في التكرار الأول سيقوم بزيادة قيمة المتغير loop ب واحد في كل مرة عند المرحلة 6 سيكون loop قيمته ب5 سيتوقف التكرار لعدم تحقق الشرط لان loop لابد أن يكون اقل من 5

في حالة ما كان الشرط اقل من أو يساوي سيتم زيادة التكرار مرة

```python

while loop <= 5:
    print("loop step {}".format(loop))
    loop+=1

```

```
loop step 0
loop step 1
loop step 2
loop step 3
loop step 4
loop step 5
```

لاحظ طباعة 6 أسطر ففي المرحلة الخامسة سيكون قيمة المتغير loop = 5 وهي تتوافق مع الشرط الذي يشترط بان تكون loop اقل من أو تساوي القيمة 5

دالة len تستخدم لمعرفة عدد العناصر الموجودة داخل الـ list

المصفوفات – Arrays

المصفوفات هي مخزن للبيانات
تحتوي المصفوفة على مجموعة من العناصر

1. كل عنصر من هذه العناصر مرقم برقم index محزن في ذاكرة الجهاز
2. الـ index قيمة فريدة لايمكن تكرارها
3. ترقيم العناصر داخل المصفوفة من اليسار لليمين
4. يبدا الترقيم من صفر مثال 0و1و2و3و4و5و6 وهكذا
5. يقصد بطول المصفوفة او حجمها عدد العناصر الموجوده بها

0	1	2	3
London	Paris	Berlin	Roma

رسم توضيحي 1مثال على مصفوفة أحادية

في الصورة السابقة مدينة Roma في الترتيب رقم 3 في حين عدد عناصر المصفوفة 4

يجب أن تفرق بين عدد عناصر المصفوفة وترقيم العناصر (index) بها فعدد العناصر هو كم عنصر تحتويه المصفوفة في مثالنا السابق 4 مدن أما الـ indexفهو موضع العنصر في المصفوفة مع ملاحظة ان الترقيم يبدا من صفر ومن جهة اليسار

قد تحتوي المصفوفة على أكثر من صف في المثال السابق المصفوفة تحتوي على صف واحد من العناصر لذلك سميت بالاتحادية.
قد تحتوي المصفوفة على أكثر من صف 3و4و5 .... الخ

تسمي المصفوفات على حسب عدد الصفوف صف واحد أحادية وصفان ثنائية وثلاثية ورباعية وخماسية ... الخ

في برنامج اكسل يحتوي البرنامج على صفوف تمثل المحور الراسي وسنرمز لها بالرمز X والأعمدة تمثل المحور الأفقي وسنرمز لها بالرمز Y.

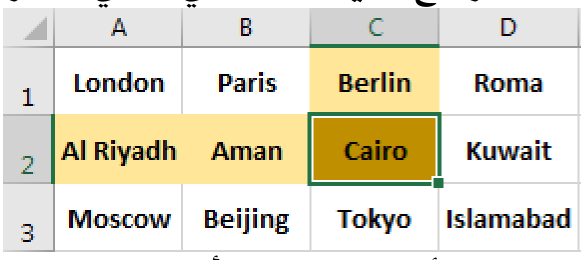
تلاقي المحور X مع Y ينتج الخلايا وهذه الخلايا هي عناصر المصفوفة **elements** وكل عنصر من هذه العناصر يمكن الوصول اليه بمعلومية موضوعة من حيث تقاطع محوري X وY

X

Y

	A	B	C	D
1	London	Paris	Berlin	Roma
2	Al Riyadh	Aman	Cairo	Kuwait
3	Moscow	Beijing	Tokyo	Islamabad

مثال موضع مدينة Cairo هي تلاقي العمود C مع الصف **2** فيكون موضعها **C2**



	A	B	C	D
1	London	Paris	Berlin	Roma
2	Al Riyadh	Aman	Cairo	Kuwait
3	Moscow	Beijing	Tokyo	Islamabad

لو قمنا بأبدال أسماء الأعمدة والصفوف بترقيم الـ **index** الخاص بالصف والعمود لكان موضعها [1,2]

	0	1	2	3
0	London	Paris	Berlin	Roma
1	Al Riyadh	Aman	Cairo	Kuwait
2	Moscow	Beijing	Tokyo	Islamabad

ملحوظة ترقيم الصفوف يبدا بصفر ومن اليسار لليمين والأعمدة بصفر ومن اعلى إلي أسفل

دالة numpay وشرح المصفوفات

قم بتثبيت مكتبة numpy إن لم تكن مثبتة راجع هذا الدرس

في المثال التالي سوف ننشئ مصفوفة بها عواصم الدول

```
#using numpy
import numpy as np

Eroup = ["London","Paris","Berlin","Roma"]
Arab  = ["Al Riyadh","Amman","Cairo","Kuwait"]
Asia  = ["Moscow","Beijing","Tokyo","Islamabad"]

Capital = np.array([Eroup,Arab,Asia])
print(Capital)
```

```
[['London' 'Paris' 'Berlin' 'Roma']
 ['Al Riyadh' 'Amman' 'Cairo' 'Kuwait']
 ['Moscow' 'Beijing' 'Tokyo' 'Islamabad']]
```

المصفوفة السابقة من النوع الثلاثي لاحتوائها على ثلاث صفوف

في المثال التالي سنقوم بطباعة عاصمه جمهورية مصر العربية Cairo بمعلومية موضعها من المصفوفة وهي في هذه الحالة [1,2]

```
#using numpy
import numpy as np

Eroup = ["London","Paris","Berlin","Roma"]
Arab  = ["Al Riyadh","Amman","Cairo","Kuwait"]
Asia  = ["Moscow","Beijing","Tokyo","Islamabad"]

Capital = np.array([Eroup,Arab,Asia])

print("Egypt Capital is {}".format(Capital[1,2]))
```

```
Egypt Capital is Cairo
```

لنقم بتغيير ترتيب المصفوفة بوضع مصفوفة Arab قبل Eroup وطباعة المصفوفة وموضع مدينة Cairo [1,2]

```
# arrays
import numpy as np

Eroup = ["London","Paris","Berlin","Roma"]
Arab  = ["Al Riyadh","Amman","Cairo","Kuwait"]
Asia  = ["Moscow","Beijing","Tokyo","Islamabad"]

Capital = np.array([Arab,Eroup,Asia])

print(Capital)
print("==================")
print("Egypt Capital is {}".format(Capital[1,2]))
```

```
[['Al Riyadh' 'Amman' 'Cairo' 'Kuwait']
['London' 'Paris' 'Berlin' 'Roma']
['Moscow' 'Beijing' 'Tokyo' 'Islamabad']]
===================
Egypt Capital is Berlin
```

1. نلاحظ تغيير ترتيب المصفوفة بان أصبح الصف الأول هو العواصم العربية وليست الأوربية
2. إن مدينة Berlin اتخذت موضع مدينة Cairo في المصفوفة

الدوال Functions

تعتبر الدوال من أساسيات أي لغة برمجية فهي تختصر في الوقت ومساحة العمل (تقليل عدد الأسطر البرمجية) وحجم البرنامج.
كل دالة قمنا ببرمجتها لها وظيفة معينة تقوم بتنفيذها ويتم تعريفها في ملف واستخدامها في نفس الملف أو باستدعائها في ملفات أخري.

> من الأفضل تعريف الدوال في ملف واستدعائها في وقت الحاجة اليه

على سبيل المثال لو نعمل على برنامج تعليمي ونريد حساب النسبة المئوية لنتيجة كل طالب. سنقوم بتكرار العملية الحسابية لكل طالب. فماذا لو قمنا ببرمجة دالة تقوم بهذه العملية الحسابية هل تتخيل كم من الوقت الذي وفرناها وعدد الأسطر البرمجية.
لو أردنا برمجة كود معقد من 10 أسطر تخيل معي كمية الأسطر البرمجية في حال تكراره كل مرة.
لو أخطأت بالعملية الحسابية في مثالنا السابق بالقسمة على درجة إجمالية خطا لتصحيح هذا الخطأ سوف تقوم بتغيير العملية الحسابية لكل طالب علي حدي لو لديك 1000 طالب هل تتخيل كم الوقت المهدر في مثل هذا التعديل البسيط.
باستخدام الدوال ستقوم بالتعديل في سطر واحد وسيتم تنفيذه في كل موضع تم استدعاء الدالة فيه

1. لكل دالة وظيفة تقوم بتنفيذها مثال دالة تقوم بحساب نسبة النجاح للطلاب.
2. يمكنك تعريف الدالة في أي موضوع في الملف.
3. في حال استدعاء الدالة من ملف خارجي يجب استدعائها قبل السطر التي تستخدم فيه ويفضل استدعاء كل الملفات اعلي الملف.
4. الدالة تؤدي وظيفة مشتركة كدالة حساب نسبة النجاح لكل طالب.

5. يقوم البرنامج بتعريف الية عمل الدالة في الذاكرة مما يسرع عمل البرنامج.
6. قد تكون القيم المرره للدالة ثابتة أو متغيرة فمثلا درجة الامتحان تختلف من طالب إلي طالب.
7. تستخدم الدالة متغيرات مختلفة الأنواع البيانات مثل **string, integer, list**... الخ.
8. يمكن للدالة إرجاع مختلف أنواع البيانات مثل **string, integer, list**... الخ. أو حتى قيمة فارغة.
9. يمكن استخدام خصائص اللغة في نطاق عمل الدالة مثل **if, loop**..... الخ.

تعرف الدالة هكذا

```
#define function

def function_name(parameter, parameter,parameter,parameter):
    #code Here

```

تعرف الدالة باستخدام كلمة **def** أي **define** يليها اسم الدالة يليها قوسان () يحتويان على كل المتغيرات ثم علامة النقطتين: ثم مجال عمل الدالة.

1. يمكن تعريف الدالة بدون استخدام المتغيرات **parameters**
2. **Parameter** أو المتغير قم يحمل أي نوع من القيمة
3. نطاق عمل الدالة (تنفيذ الكود البرمجي) يبدا في السطر التالي لتعريف الدالة وعلامة: مع مسافة بادئة لليسار

يتم استخدام الدالة بكتابة اسمها ملحق بقوسين () مع كتابة قيم المتغيرات إن وجدت.

مثال قم بكتابة دالة تطبع جملة **I am Python** أين قراءة هذه الجملة مسبقا ؟؟؟؟

```
1  #define function
2  
3  def print_hello_message():
4      print("Hello I am Python")
5  
6  print_hello_message()
7  
```

```
Hello I am Python
```

1. في ال سطر3 قمنا بتعريف الدالة وأعطيناها اسم **print_hello_message** أي طباعة رسالة ترحيبية.
2. في ال سطر4 قمنا بتنفيذ كود طباعة الجملة **hello I am Python** لاحظ المسافة البادئة من اليسار
3. في ال سطر6 قمنا باستدعاء الدالة.

استخدام المتغيرات parameters

لو أردنا تغيير كلمة **Python** إلى كلمة أخرى

```
#define function with parameters

def print_hello_meassage(name):
    print("Hello I am {}".format(name))
print_hello_meassage("Java")
print_hello_meassage("Php")
```

```
Hello I am Java
Hello I am Php
```

إضافة المتغير بقيمة بادئة ثابتة

لو قمنا باستدعاء الدالة بدون إعطاء المتغير **name** قيمة سيظهر لنا البرنامج خطأ مفاده لا يمكن استدعاء الدالة بدون إعطاء متغيراتها قيمة.

```
#define function with parameters

def print_hello_meassage(name):
    print("Hello I am {}".format(name))
print_hello_meassage()

```

```
Traceback (most recent call last):
    print_hello_meassage()
TypeError: print_hello_meassage() missing 1 required positional argument: 'name'
```

مثال 1:

```
#define function with parameters

def print_hello_meassage(name=""):
    print("Hello I am {}".format(name))
print_hello_meassage()

```

```
Hello I am
```

مثال 2:

```
#define function with parameters

def print_hello_meassage(name="Python"):
    print("Hello I am {}".format(name))
print_hello_meassage()

```

```
Hello I am Python
```

يمكنك إضافة الـparameter دون التقيد بموضعه من المتغيرات الأخرى بكتابة اسمه = القيمة مثال"name="Python

استخدام return

تستخدم return لتحديد القيمة التي سترجعها الدالة

عند استخدام return فلن تدخل الأسطر التي تليها ضمن نطاق عمل الدالة (return تعني نهاية عمل الدالة)

في مثالنا التالي سوف ننشئ دالة تقوم بإرجاع النسبة المئوية لنتيجة طالب

```
#using return in function

def final_score(degree):
    return (degree*100/300)

_degree = final_score(250)
print("Student Degree is {} %".format(_degree))
```

```
Student Degree is 83.33333333333333 %
```

لنطور دالتنا السابقة لتطبع التقدير النهائي للطالب حسب الجدول التالي

التقدير	الوصف	النسبة المئوية
A	جيد جدا	90 ٪ -100 ٪
B	فوق المتوسط	80 ٪ -89 ٪
C	متوسط	70 ٪ -79 ٪
D	دون المتوسط	60 ٪ -69 ٪
F	راسب	0

```
#using if in function

my_degree = 250

def final_score(degree):
    final = (degree*100/300)
    if final >= 90:
        return "A"
    elif final >= 80:
        return "B"
    elif final >= 70:
        return "C"
    elif final >= 60:
        return "D"
    else:
        return "F"

_degree = final_score(my_degree)

print("Student Degree is {} Score is {}".format(my_degree,_degree))
print("Student Degree is 20 Score is {}".format(final_score(20)))
```

```
Student Degree is 250 Score is B
Student Degree is 20 Score is F
```

لنطبع تقييم الطالب بالعربية والإنجليزية

```python
#using if in function

arabic_score = {"A":"جيد جدا","B":"فوق المتوسط","C":"متوسط","D":"دون المتوسط","F":"راسب"}
my_degree = 250

def final_score(degree):
    final = (degree*100/300)
    if final >= 90:
        return "A"
    elif final >= 80:
        return "B"
    elif final >= 70:
        return "C"
    elif final >= 60:
        return "D"
    else:
        return "F"

_degree = final_score(my_degree)

print("Student Degree is {} Score is {} {}".format(my_degree,_degree,arabic_score[_degree]))
_degree = final_score(20);
print("Student Degree is 20 Score is {} {}".format(_degree,arabic_score[_degree]))
```

```
Student Degree is 250 Score is B فوق المتوسط
Student Degree is 20 Score is F راسب
```

استخدام متغير خارجي داخل الدالة

قد تحتاج إلى استخدام متغير واحد في أكثر من دالة نستخدم كلمة global قبل اسم المتغير داخل الدالة

```
#using global var

arabic_score = {"A":"جيد جدا","B":"فوق المتوسط","C":"متوسط","D":"دون المتوسط","F":"راسب"}
my_degree = 250

def final_score():
    global my_degree

    final = (my_degree*100/300)
    if final >= 90:
        return "A"
    elif final >= 80:
        return "B"
    elif final >= 70:
        return "C"
    elif final >= 60:
        return "D"
    else:
        return "F"

_degree = final_score()
print("Student Degree is {} Score is {} {}".format(my_degree,_degree,arabic_score[_degree]))
```

```
Student Degree is 250 Score is B فوق المتوسط
```

لاحظ قمنا بحذف متغير degree وقمنا باستدعاء متغير my_degree من خارج الدالة باستخدام كلمة global

استدعاء داله داخل داله

```python
#call function into function

def calculation_degree(_degree):
    return (_degree*100/300)

def print_degree(my_degree):
    final_degree = calculation_degree(my_degree)
    print("student degree is {} ".format(final_degree));
print_degree(200)
```

```
Student Degree is 66.66666666666667
```

دالة calculation_degree المعلمة باللون الأحمر قمنا باستدعائها داخل دالة print_degree

Lambda

تعمل عمل الدالة function فهي عبارة عن تعريف دالة في سطر واحد

1. **يمكن إضافة أكثر من متغير parameters**
2. **تقوم بإخراج عنصر واحد فقط في سطر واحد فقط (مجال عملها سطر واحد فقط)**
3. **يمكن التعبير هنا بمتغير**

يتم كتابتها بهذا الشكل

```python
lambda arguments : expression
```

```python
#define lambda

score = lambda degree, total: (degree*100)/total
y_score = round(score(20,300),2);
print("Your score is {}%".format(y_score))
```

```
Your score is 6.67%
```

دالة round تعرض عدد محدد من الأرقام العشرية في حال لم تضع قيمة فستقرب لأقرب رقم عشري

Try & Except

تستخدم لتفادي الأخطاء المنطقية الخاصة بالتعامل مع أساسيات اللغة.

أثناء العمل على برنامجك الخاص قد يكون هناك خطأ منطقي من النظام كطباعة متغير غير موجود أو استدعاء الدالة من دون متغيراتها

```
#Try & Except
print(name)
```

```
Traceback (most recent call last):
    print(name)
NameError: name 'name' is not defined
```

خطا كهذا أثناء عمل البرنامج سيوقفه عن العمل مما يؤدي إلي خسارتك المزيد من العملاء أو ثقة مديريك.

لاحظ NameError هو نوع الخطأ

1. **try مساحة اختبار الخطأ.**
2. **except مساحة العمل في حالة حدوث الخطأ.**
3. **finally سيتم تنفيذ هذ الكود في النهاية في حالة حدوث الخطأيمكنك استخدام أكثر من except.**
4. **else تستخدم في حالة عدم حدوث خطا.**

سنولي finally مزيدا من الشرح عند شرح التعامل مع الملفات

```
#Try & Except
try:
    print(name)
except:
    print("you must insert your name")
finally:
    print("this is final code")
```

```
you must insert your name
this is final code
```

في حالة توقعك نوع الخطأ يمكنك استخدامه في سطر except

```
#Try & Except
try:
 print(name)
except NameError:
    print("you must insert your name")
except:
    print("except number 2")
finally:
    print("this is final code")

```

```
you must insert your name
this is final code
```

لاحظ لم يتم طباعة الاستثناء الثاني لأنه تحقق من حدوث الخطأ الأول وهو NameError

يمكنك جمع أكثر من استثناء في try واحدة على سبيل المثال سنحاول في مثالنا السابق القسمة على صفر مع إصلاح خطأ NameError

```
#Try & Except
name = "Python"
try:
    print(name)
    print(10/0)
except NameError:
    print("you must insert your name")
except ZeroDivisionError:
    print("You can't division by zero")
finally:
    print("your name is {}".format(name))

```

```
Python
You can't division by zero
your name is Python
```

لاحظ تخطي خطا **NameError** فوجد خطا **ZeroDivisionError** فقام بتنفيذ الاستثناء وفي النهاية قام بطاعة اسمك.
تستخدم **else** في حالة عدم وجود أي استثناء.

```
#Try & Except
name = "Python"
division = 10
try:
    print(name)
    print("Result is {}".format(10/division))
except NameError:
    print("you must insert your name")
except ZeroDivisionError:
    print("You can't division by zero")
else:
    print("you division by {}".format(division))

```

```
Python
Result is 1.0
you division by 10
```

للمزيد عن هذا لموضوع تابع هذا الرابط
https://docs.python.org/3/tutorial/errors.html

التعامل مع الملفات

هناك امرين يجب أن تعلمهم جيدا في حالة تعاملك مع الملفات في أي لغة البرمجة

1. Extension امتداد الملف

يقصد به نوع الملف وهو اختصار يلي رمز النقطة في أخر الاسم مثال filename.txt فنوع الملف هنا txtويشير هذا الرمز إلي أن الملف عبارة عن نص مثل هذه الامتدادات

pdf, docx,psd ...etc. للمزيد حول هذا الموضوع
https://en.wikipedia.org/wiki/Filename_extension

2. Path مسار الملف

ويقصد به مكان الملف في المساحة التخزينية التي تستخدمها (hard disk) أو مساحة العمل للمزيد
https://en.wikipedia.org/wiki/Path_(computing)

س ما هو نوع الملف الذي ينتهي ب .py ؟

بعض الدوال والأوامر المهمة في التعامل مع الملفات

الدالة	المعنى	الاستخدام
open	فتح	لفتح ملف سواء للقراءة أو الكتابة
"r"	----	ترمز لاستخدام الملف للقراءة
"a"	----	للإضافة نص جديد إلي النص الموجود بالملف
"w"	----	لاستخدام الملف للكتابة
"x"	----	لإنشاء ملف جديد
"t"	----	لجعل الملف نوعه نصي **Text**
"b"	----	لجعل الملف نوعه نصي **Binary**
read()	قراءة المحتوي	تستخدم لقراءة محتويات الملف
readline()	قراءة سطر	لقراءة سطر في الملف لقراءة كامل الملف استخدمها في **loop**
write()	كتابة	للكتابة في الملف لابد أن يكون الملف فتح للكتابة "**W**"
remove()	حذف	حذف الملف
exists()	موجود	للتأكد من وجود الملف تقوم برجاع **Boolean**
rmdir()		لحذف مجلد

إنشاء ملف

```
#working with files

file = ""
try:
    file = open("my_score.txt","x")
except:
    print("File not created")
finally:
    if file:
        print("File created")
```

```
File created
```

حاول تنفيذ السطر من جديد ستكون النتيجة **File not created** هذا منطقي لان الملف منشئ بالفعل فلا يمكن إنشاء ملفين بنفس الاسم

لذ وجب علينا التأكد إذا ما كان الملف موجود مسبقا ام لا باستخدام دالة **exists**

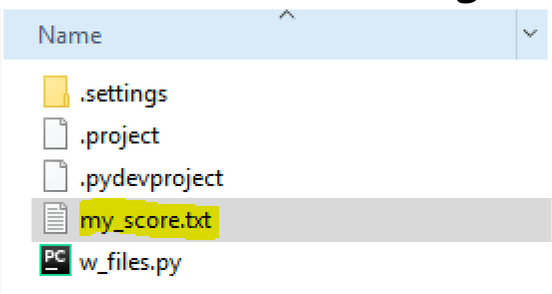
```
#working with files
import os

file = ""
filename= "my_score.txt"
try:
    if os.path.exists(filename) == False:
        file = open(filename,"x")
except:
    print("File not created")
finally:
    if file:
        print("File created")
```

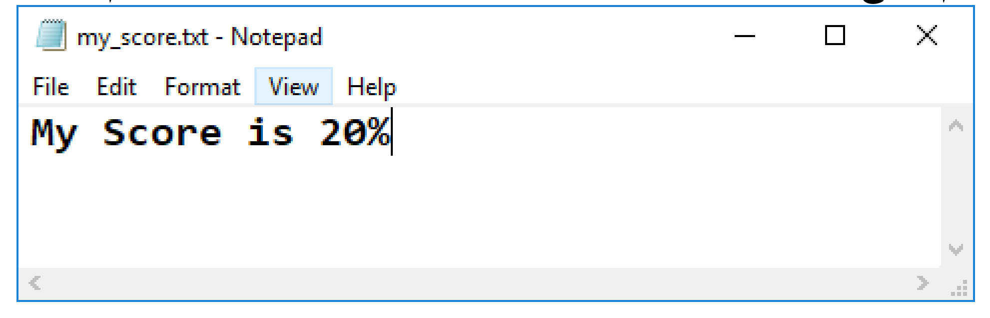
1. في السطر 2 قمنا باستدعاء مكتبة **os** لاستخدمها في التأكد من وجود الملف مع **path**
2. في السطر 5 قمنا بنشاء متغير يحمل اسم الملف المراد لأنشائه لأننا سنستخدمه في أكثر من موضع
3. في السطر 7 قمنا بالتأكد من وجود الملف باستخدام دالة **exists** الموزثة من ملف **os. path** التي تقوم برجاع **True** اذا كان الملف موجود و **False** اذا كان غير موجود
4. في السطر 8 نقوم بإنشاء الملف

> في حالة استخدام قيمة واحدة في أكثر من موضع في البرنامج يفضل تعريفها في متغير لسهولة تعديلها في المستقبل

قم بالذهاب إلى المجلد الخاص بالمشروع سوف تجد اللمف موجود ضمن ملفات المشروع



قم بفتح الملف بمحرر النصوص واكتب فيه **My Score is 20%** ثم اغلق النافذة



قراءة محتويات ملف

```
#working with files
import os

file = ""
filename= "my_score.txt"
try:
        file = open(filename,"r")
        print("file content is: {}".format(file.read()))
except:
    print("Error happened while reading file")
```

```
file content is: My Score is 20%
```

تعديل ملف

هناك طريقتين للتعديل على الملف

1. الإضافة إلى المحتوي الموجود من قبل نستخدم "a" مع دالة open

```
#working with files
import os

file = ""
filename= "my_score.txt"
try:
        file = open(filename,"a")
        file.write("\n new content to exists file");

        file = open(filename,"r")
        print("file content is: {}".format(file.read()))
except:
    print("Error happened while reading file")
```

```
file content is: My Score is 20%
 new content to exists file
```



تستخدم \n للنزول سطر جديد

2. استبدال المحتوي بالمحتوي الجديد نستخدم "w" مع دالة open

```
#working with files
import os

file = ""
filename= "my_score.txt"
try:
        file = open(filename,"w")
        file.write("new content to exists file");

        file = open(filename,"r")
        print("file content is: {}".format(file.read()))
except:
    print("Error happened while reading file")

```

```
file content is: new content to exists file
```

حذف ملف

لحذف ملف نستخدم دالة remove الموجودة في مكتبة os يجب التأكد من وجود الملف أولا قبل الحذف

```
#working with files
import os

file = ""
filename= "my_score.txt"
try:
    remov_file = os.remove(filename)
except:
    print("Error happened while reading file")

```

مثال متقدم على التعامل مع الملفات

يعتبر هذا المثال تدريب على ما سبق شرحه

سنقوم بإنشاء دالة تقوم بإرجاع التقدير النهائي للثلاث طلاب وسنقوم بإنشاء ملف لكل طالب ونسجل فيه النسبة المئوية لتقديره

```
#working with files
import os

#define vars
names   = ["Sayed","Ahmed","Ali","Mustafa"]
degrees = [250,300,100,30]

message = "{} Your Score is: {}%"
ex = ".txt"
i = 0 #will increase value with 1 in loop

#define function calculate score
def final_degree(degree=0):
    return degree*100/300

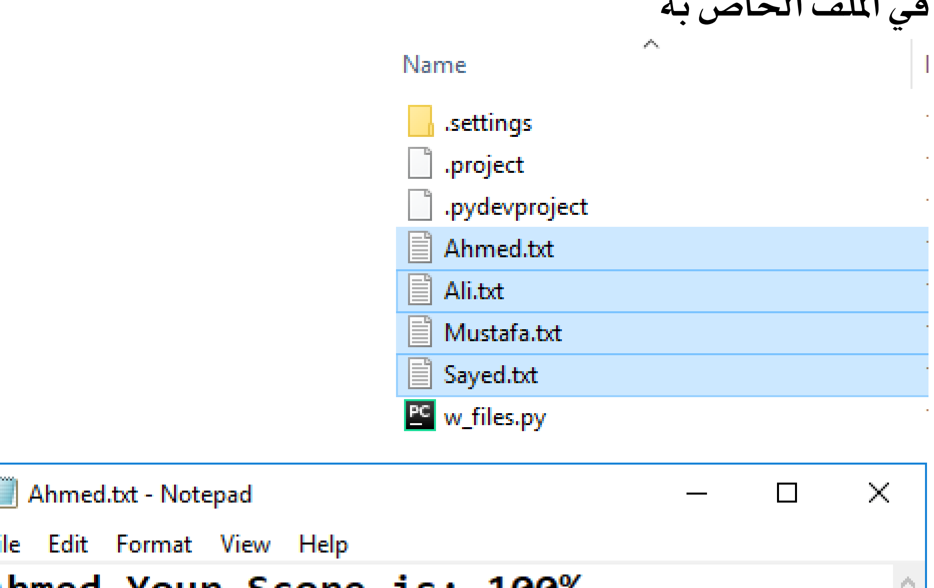

def write_in_file(filename):
    global i
    global message
    if os.path.exists(filename) == False:
        #create the file if not exists
        open(filename,"x")

    #write into file
    file = open(filename,"w")
    fnl_degree = round(final_degree(degrees[i]))
    file.write(message.format(name,fnl_degree))
    i+=1


#loop all names
for name in names:
    filename = name+ex
    write_in_file(filename)

```

بالذهاب إلى مجلد البرنامج تجد أن الملفات قد أنشئت وتم إضافة درجة كل طالب في الملف الخاص به



التعامل مع المجلدات

تعتبر مكتبة os متخصصة في التعامل مع المجلدات

لتتبع المجلد داخل المسار يستخدم علامتي\\ للفصل بين كل مجلد ومجلد
C:\\Users\\Dell\\eclipse-workspace\\hello

لا يجاد المجلد الحالي دالة getcwd

```
#work with folders
import os
print(os.getcwd())
```

```
C:\Users\Dell\eclipse-workspace\hello
```

```
#work with folders
import os
os.chdir('C:\\Users\\Dell\\eclipse-workspace')
print(os.getcwd())
```

```
C:\Users\Dell\eclipse-workspace
```

لتغير المسارchdir

عرض قائمة بالملفات داخل المجلدlistdir

```
#work with folders
import os
print(os.getcwd())
print(os.listdir())

```

```
C:\Users\Dell\eclipse-workspace\hello
['.project', '.pydevproject', '.settings', 'constant.py',
'database_module.py', 'index.py', 'w_files.py', '__pycache__']
```

إنشاء مجلد جديد mkdir

```python
#work with folders
import os
os.mkdir('test')
print(os.listdir())
```

إعادة تسمية ملف داجل المجلد rename

```python
#work with folders
import os
os.rename('test',"test_new")
print(os.listdir())
```

حذف ملف داخل المجلد rmdir()

```python
#work with folders
import os
os.rmdir("test_new")
print(os.listdir())
```

```
['.project', '.pydevproject', '.settings', 'constant.py', 'database_module.py', 'index.py', 'w_files.py', '__pycache__']
```

قمنا بحذف مجلد test_new بدلا من test لأننا قمنا تغيير اسم test إلى test_new دالة rmdir تقوم بإلغاء المجلدات الفارغة فقط إذا كان المجلد غير فارغ يجب عليك تفريغه أولا قبل استخدام دالة الحذف

أنشئ مديولك الخاص

المديول هو عبارة ملف تقوم بإنشائه في برنامجك يحتوي على كود برمجي قمت ببرمجته مسبقا لتنفيذ مهام محددة

مثلا تقوم بالعمل على برنامج مكتبة الكرتونية تتعامل مع قواعد البيانات مثل عمليات اتصال وإدخال وتعديل وحذف واستعراض بيانات فتحتاج إلي إنشاء أقسام وإدراج كتب وتحديث بيانات كتاب أو إدراجه في قائمة الكتب المستعارة.

فليس عملياً كتابة أسطر الاتصال بالقاعدة والعمليات في كل صفحة أو عند كل عملية.

هذا لا يعنى أن برنامجك لن يعمل ولكن تخيل معي أن بيانات الاتصال بقاعدة البيانات تم تغييرها فستضطر إلى البحث في كل الملفات التي بها اتصال بقاعدة البيانات للتعديل على بيانات الاتصال تخيل معي قدر الوقت والجهد المبذول علاوة على ذلك عدد الأسطر البرمجية قد تصل إلي المئات من السطور في حين يمكن اختصاره باقل من ذلك.

المبرمج المتميز من يقوم ببناء الكود البرمجي بغية تطويره وليس مجرد كود يعمل.

سنقوم بإنشاء ملف باسم database_module.py وهو الخاص بالتعامل مع قاعدة البيانات

.

```
#Database module

"""===============================
        connect to database
==================================="""
def connect_to_db(dbname="test",dbusername="root",dbpassword="123456",dbhost="localhost"):
    print("database connected successfully with information dbname:{} dbusername:{} dbpassword:{} dbhost:{} ".format(dbname,dbusername,dbpassword,dbhost))

"""===============================
        insert into database
==================================="""
def insert_into_db(query=""):
    try:
        print("insert query {} success".format(query))
    except:
        print("insert error")

"""===============================
        update database
==================================="""
def update_db(query=""):

"""===============================
        delete database
==================================="""
def delete_db(query=""):
    try:
        print("delete query {} success".format(query))
    except:
        print("delete error")

"""===============================
        select database
this function will select data from database
params: query
return: string
```

```
====================================="""
def select_db(query=""):
    try:
        print("select query {} success".format(query))
    except:
       print("select error")
```

سنقوم بإنشاء ملف باسم **index.py** ونقوم باستدعاء مديول **database_module** فيه نقول بعمليات مختلفة على قاعدة البيانات.

```
import database_module as db

db.connect_to_db("library","admin","admin", "localhost")

#insert new book
db.insert_into_db("insert  into  table  books  bookname='learn python'")

#update bookname learn python
db.update_db("update table books bookname='learn python edition 2' where bookname='learn python' ")

#delete bookname learn python
db.delete_db("delete  from  table  books  where  bookname='learn python' ")

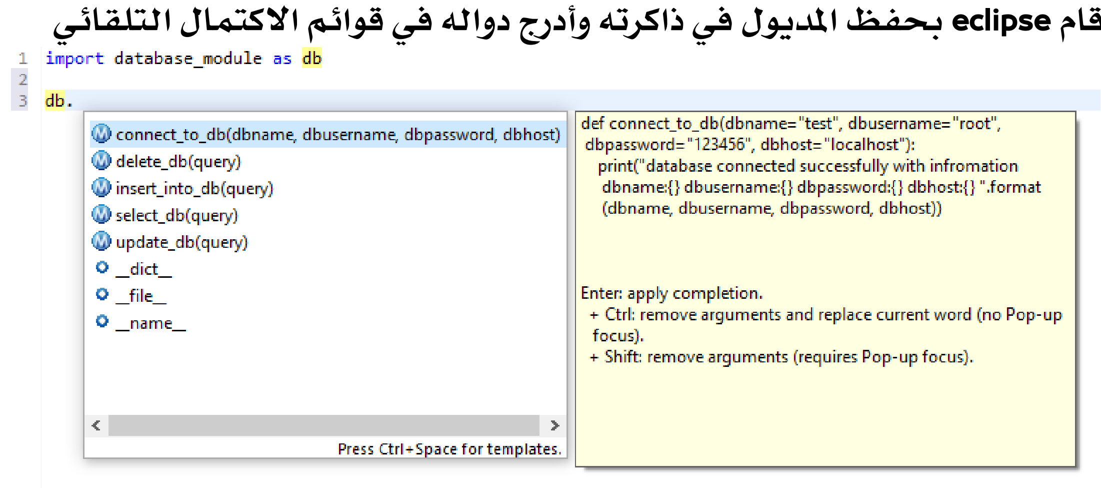
#select books learn python
db.select_db("select * from table books")

```

```
database  connected  successfully  with  information  dbname:  library
dbusername: admin dbpassword: admin dbhost:localhost
insert query insert into table books bookname='learn python' success
updated query update table books bookname='learn python edition 2' where
bookname='learn python' success
delete  query  delete  from  table  books  where  bookname='learn python'
success
select query select * from table books success
```

هذا مثال تبسيطي للاتصال بقواعد البيانات في الحقيقة التعامل مع قواعد البيانات غير ذلك

قام **eclipse** بحفظ المديول في ذاكرته وأدرج دواله في قوائم الاكتمال التلقائي



من الأشياء الرائعة في **eclipse** النوافذ التوضيحية لاحظ هنا اظهر المتغيرات

```
import database_module as db
                  dbname="test", dbusername="root", dbpassword="123456", dbhost="localhost"
db.connect_to_db(dbname, dbusername, dbpassword, dbhost)
```

الخاصة بالدالة مع القيم الافتراضية لها

التعامل مع الوقت datetime

مكتبة datetime هي المتخصصة في التعامل مع الوقت والتاريخ

مثال طباعة الوقت الحالي

```
#date and time
import datetime

date_now = datetime.datetime.now()

print("date now is {}".format(date_now))

print("Year is {}".format(date_now.year)
print("day is {}".format(date_now.day))
```

```
date now is 2018-12-07 18:12:12.643742
Year is 2018
day is 7
```

تم استخراج التاريخ بصيغة سنة، شهر، يوم، ساعة، دقيقة، ثانية، وجزء من الثانية

التاريخ يقرا من اليسار لليمين

ماذا لو أردنا معرفة اسم اليوم الحالي

تستخدم دالة strftime لاستخراج صيغ مختلفة للتاريخ انظر جدول الرموز الخاصة بالتاريخ

```
#date and time
import datetime

date_now = datetime.datetime.now()

print("This day is {}".format(date_now.strftime("%A")))
```

```
This day is Friday
```

جدول رموز الوقت والتاريخ

الرمز	الوصف	النتيجة
%a	أول 3 احرف من اسم اليوم في الأسبوع	Mon
%A	اسم اليوم بالكامل	Monday
%w	موضع اليوم بالنسبة للأسبوع يبدا الأسبوع من يوم الأحد = 0 الاثنين = 1 الثلاثاء = 3 وكهذا	1
%d	عدد أيام الشهر	30
%-d	عدد الأيام في الشهر بالأرقام العشرية إن وجدت	30
%b	أول ثلاث احرف من اسم الشهر	Sep
%B	اسم الشهر بالكامل	September
%m	رقم الشهر بالنسبة للسنة مسبوق بصفر في الأرقام من 1-9	09
%-m	رقم الشهر صحيح بدون صفر	9
%y	السنة مع حذف الرقمين الأولين في مثالنا هذا 2018	18
%Y	السنة بالأربعة أرقام كاملة	2018
%H	الساعة بنظام 24 ساعة	22
%-H	الساعة بنظام 12 ساعة	7
%I	الساعة بنظام 12 ساعة مسبوقة بصفر في الأرقام من 1-9	07
%-I	الساعة بنظام 12 ساعة بدون صفر	7
%p	تحديد وقت الساعة صباحا AM ومساءً PM	AM
%M	الدقائق بصفر بادي مع الأرقام من 1-9	06
%-M	الدقيقة بدون صفر	6
%S	الثانية مسبوقة بصفر في الأرقام من 1-9	05
%-S	الثانية بدون صفر	5
%f	جزء من الثانية مسبوق بـ00	000000

%z	نظام UTC الخاص بالزيادة والنقصان عن توقيت جرينتش مثال مصر قبل توقيت جرينتش ب ساعتين بضرب عدد الساعات في عدد الدقائق في كل ساعة 2*60=120	+0120
%Z	المنطقة الزمنية للمزيد https://www.timeanddate.com/time/zones/cst	CST
%j	موضوع اليوم في السنة بصفر في البداية مع الأرقام من 1-9	273
%-j	موضع اليوم في السنة بدون صفر في البداية	273
%U	موضوع الأسبوع في السنة مع اعتبار بداية الأسبوع من يوم الأحد	39
%W	موضع الأسبوع في السنة مع اعتبار يوم الاثنين بداية الأسبوع	39
%c	الوقت الحالي اسم اليوم والشهر وتاريخ اليوم والسنة والساعات والدقائق والثواني	Mon Sep 30 07:06:05 2013
%x	التاريخ اليوم والشهر والسنة مع بداية صفر في الأرقام من 1-9	09/30/13
%X	الساعة والدقيقة والثانية مع بداية صفر في الأرقام من 1-9	07:06:05

الفصل الثاني تطبيق عملي

في هذا الفصل سنطبق تطبقا عمليا لكل ما درسناه في الفصل الأول سنقوم بتنفيذ أله حاسبة بسيطة تحسب زكاة المال الواجبة بإدخال المبلغ الإجمالي المستحق الزكاة وعند ضغط زر احسب الزكاة تظهر على الشاشة قيمة الزكاة واجبة السداد.

ستتعلم مراحل تصميم البرنامج:

1. التحليل
2. التصميم
3. كتابة الكود البرمجي

كيفية تصدير برنامج بصيغة EXE لكي تتمكن من نشرة على الأنترنت أو نسخة على أسطوانات وتوزيعه كما ستتمكن من تغير أيقونة البرنامج إلى أيقونة خاصة بك.

مراحل تنفيذ برنامج

يتم تنفيذ البرامج على عدة مراحل متتالية وهي

1. التحليل: في هذه المرحلة يتم تحديد ماهية البرنامج وتحديد المهام المطلوبة في البرنامج عن طريق لغة UML وليس هنا المجال للحديث عن هذه اللغة.
2. التصميم: تبدأ بعد مرحلة التحليل يقوم المصمم بتصميم wireframe (كروكي) لشاشات البرنامج وعرضة على فريق التحليل للتأكد من تماشيه مع فكرة البرنامج العامة بعد الموافقة يشرع المصمم في العمل على التصميم.
3. كتابة الكود البرمجي: قد تبدأ بعد التحليل أو مع التصميم حسب ماهية العمل وهنا يبدا المبرمج أو فريق البرمجة في كتابة الكود البرمجي ثم ربط التصميم مع الكود ليخرج البرنامج إلي النور.
4. الاختبار: يقوم فريق الاختبار بتجربة البرنامج لرصد أي أخطاء قد تظهر أثناء العمل علية.

الخطاء من قبل المستخدم كإدخال بيانات نصية بدلا من الرقمية أو خطا في تنفيذ العمليات.

تتم تجميع الأخطاء وتسليمها للمحلل لإعادة توجيها إلي القسم المختص لحلها.

أولا: تحليل بسيط للبرنامج

1. البرنامج يستهدف المسلمين
2. يقوم البرنامج بعمليات حسابية رقمية
3. قد تكون المخرجات رقم عشري
4. تحسب نسبة زكاة المال لكل حول (عام) المبلغ قسمة 40
5. يشبه البرنامج في شكله الاله الحاسبة من حيث الشاشة والأزرار
6. عند الضغط على زر احسب زكاتك يقوم البرنامج بحساب الزكاة
7. مبلغ النصاب 100 وهو الحد التي تجب علية الزكاة
8. يمكن للمستخدم حساب الزكاة أكثر من مرة
9. يمكن للمستخدم التراجع عن أخر رقم تم إدخاله

ثانيا: التصميم

بناءً على ما تقدم يستنتج المصمم أن البرنامج عبارة عن مستطيل بأعلى الشاشة ثم يله 10 ازار من 0 إلى 9 من اجل إدخال رقم مبلغ الزكاة وزر واضح لحساب الزكاة وزر لإعادة التهيئة.

لتصميم ال wireframe يوجد العديد من البرامج والمواقع المجانية والمدفوعة لتنفيذه أو يمكنك رسمة يدويا

سوف نستخدم في مثالنا هذا موقع https://wireframe.cc كونه مجاني و أون لاين وبسيط في حالة المشاريع الكبيرة يمكنك استخدام برامج متخصصة.

ثالثا: تنفيذ الكود البرمجي

الواجهة الرسومية للمستخدم

GUI: Graphical User Interface

هي كل ما يراه المستخدم ويتفاعل معه محتوي على (العناصر الرسومية) من جداول وصور وأزار مثلا برنامج الورد عبارة عن شاشة رسومية من خلاله يمكنك إدراج صور ونماذج وجداول وعناصر مختلفة.

الGUI الخاصة بلغة البايثون عبارة عن مكتبات (سبق التحدث عن المكتبات من قبل) بها عدة دوال عند استدعائها تنفذ أوامر معينة مثلا إنشاء شاشة ويندوز أو إنشاء زر أو إدراج نص أو صورة أو **forms** مختلفة.
يوجد العديد من المكتبات أو **packages** مختصة في **GUI** ولكننا في مثالنا هذا سنستخدم مكتبة **tkinter**

1) قم بتثبيتها على برنامج الـ**eclipse** كما شرحنا سابقاً في درس تثبيت المكتبات

2) قم بفتح مشروع جديد في برنامج **eclipse** وسمة **zakat**

سنقوم بتقسيم العمل على البرنامج على ثلاث مراحل

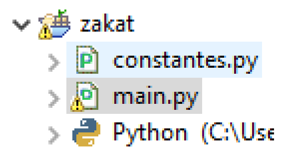
المرحلة الأولي تجهيز الـelements

في هذه المرحلة سنقوم بتعريف العناصر المطلوبة برمجيا مثل الإزار والشاشة الرئيسية ...الخ

قم بإنشاء ملف جديد باسم **constantes.py** هذا الملف سيحتوي على جميع مسميات البرنامج من الاسم إلي أسماء الإزار ... الخ سوف نستدعيه في الملف الأساسي للبرنامج. اكتب فيه الكود التالي

```
PROGTITLE = "Zakat Calculator"
BUTTONCALTEXT ="Calculate"
BUTTONCLEAR = "Clear"
BUTTONZERO ="0"
BUTTONONE ="1"
BUTTONTOW ="2"
BUTTONTHREE ="3"
BUTTONFOUR ="4"
BUTTONFIVE ="5"
BUTTONSIX ="6"
BUTTONSEVIEN ="7"
BUTTONEIGHT ="8"
BUTTONNINE ="9"
```

قم بإنشاء ملف جديد باسم **main.py** هذا هو الملف الأساسي للبرنامج ليصبح مشروعك بهذا الشكل

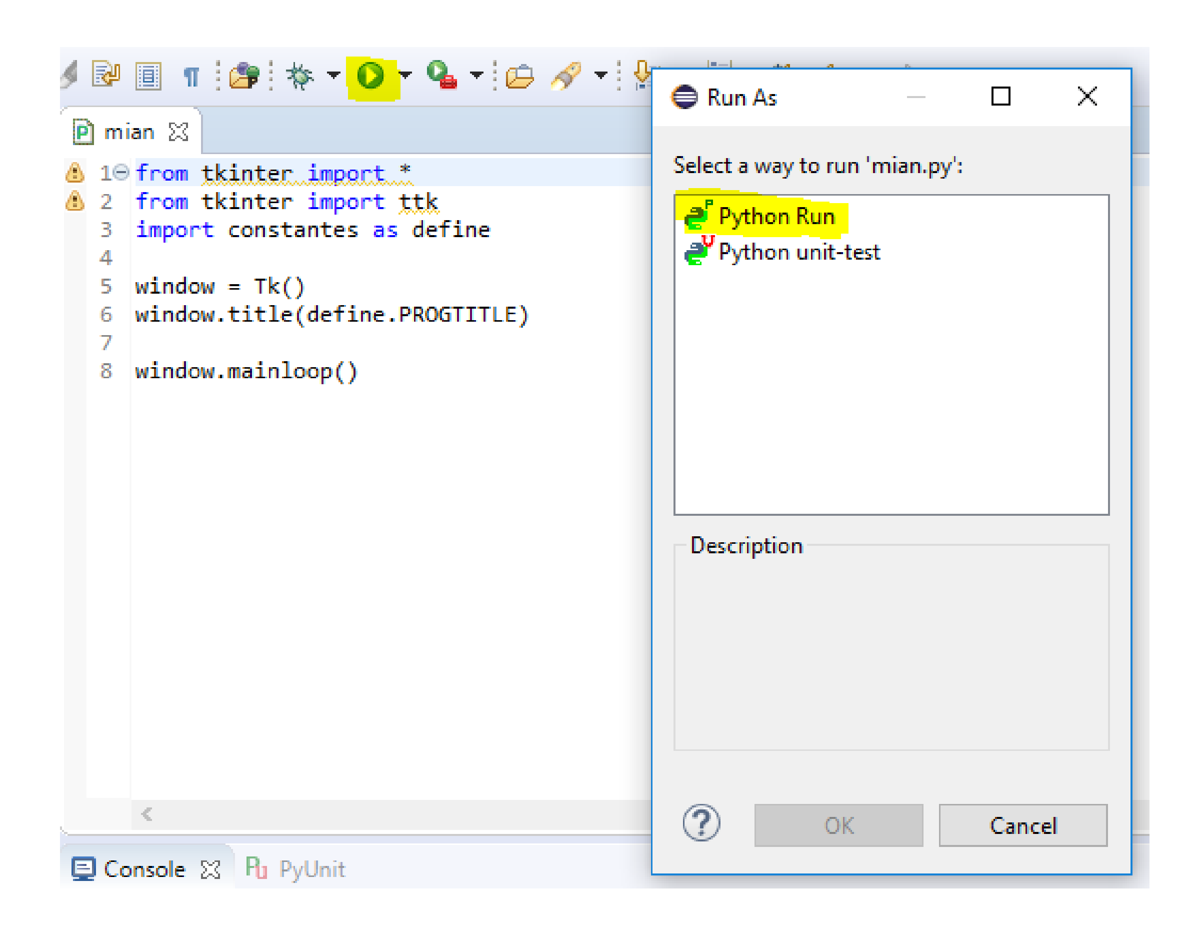


ضع الكود التالي في ملف **main.py**

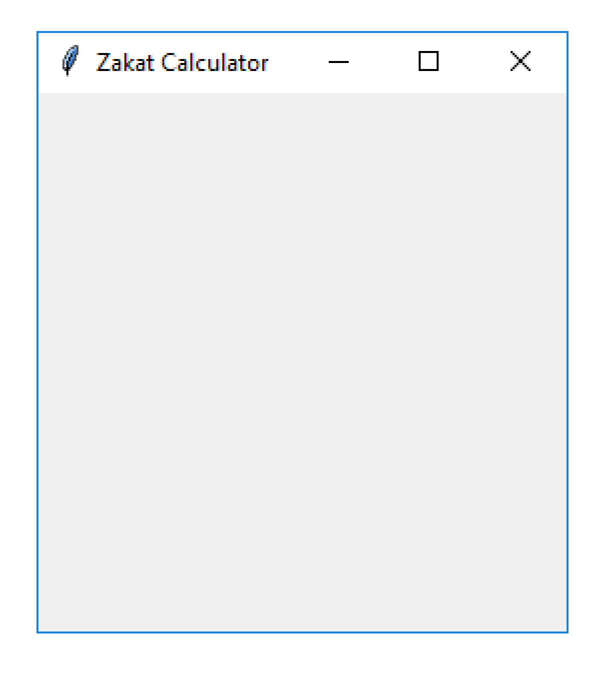
```
1 from tkinter import *
2 from tkinter import ttk
3 import constantes as define
4
5 window = Tk()
6 window.title(define.PROGTITLE)
7 window.mainloop()
```

1. من سطر **1** إلي **3** قمنا باستدعاء المكتبات المطلوبة والخاصة بال **GUI** وكذلك ملف المسميات الخاص بنا
2. في سطر **5** أعطينا متغير **window** قيمه كلاس **Tk()**
3. في سطر **6** أعطينا للبرنامج عنوان مجلوب من ملف الثوابت الذي استدعينا في سطر**3**
4. في السطر **7** استدعينا دالة **mainloop** والخاصة بإنشاء النافذة الرئيسية للبرنامج

قم بعمل **run** للبرنامج بالضغط على الزر**run** الأخضر في برنامج **eclipse** أو اضغط على زر **Ctrl+F11** من لوحة المفاتيح ثم اختار **Python Run** ثم اضغط **OK**



ستظهر لك نافذة البرنامج فارغة قم بتوسيعها من أحد أطرافها ولاحظ العنوان



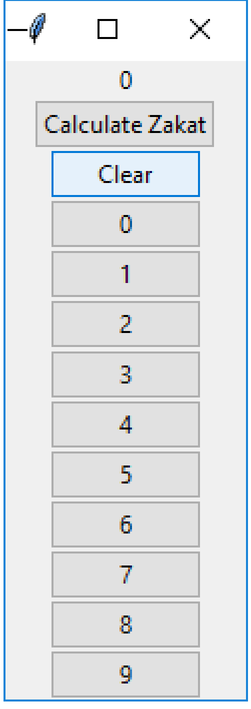
لدينا الأن شاشة برنامج لا تحتوي على أي عناصر لنقم الأن بتعريف العناصر لتظهر على الشاشة

1) نحتاج إلى عنصر Label لا ظهار الأرقام المدخلة ونتيجة العملية الحسابية
2) وعدد 12 زر الأرقام من 0 إلى 9 وزرclear لتصفير العملية الحسابية ولبداية عملية جديده وزر حساب الزكاة.

```
1   from tkinter import *
2   from tkinter import ttk
3   import constantes as define
4
5   window = Tk()
6   window.title(define.PROGTITLE)
7
8   screen_label = ttk.Label(text="0").pack()
9
10  btn_calcu = ttk.Button(text=define.BUTTONCALTEXT).pack()
11  btn_clear = ttk.Button(text=define.BUTTONCLEAR).pack()
12  btn_zero = ttk.Button(text=define.BUTTONZERO).pack()
13  btn_one = ttk.Button(text=define.BUTTONONE).pack()
14  btn_tow = ttk.Button(text=define.BUTTONTOW).pack()
15  btn_three = ttk.Button(text=define.BUTTONTHREE).pack()
16  btn_four = ttk.Button(text=define.BUTTONFOUR).pack()
17  btn_five = ttk.Button(text=define.BUTTONFIVE).pack()
18  btn_six = ttk.Button(text=define.BUTTONSIX).pack()
19  btn_sevin = ttk.Button(text=define.BUTTONSEVIEN).pack()
20  btn_egiht = ttk.Button(text=define.BUTTONEIGHT).pack()
21  btn_nine = ttk.Button(text=define.BUTTONNINE).pack()
22  window.mainloop()
```

1. **السطر 7 قمنا بتعريف متغير يحمل قيمة الـlabel وأعطينا النص الافتراضي ب 0 من خلال إعطاء قيمة صفر لـ text="0" parameter**
2. **دالة pack هي المسئولة عن إضافة العنصر إلى الشاشة الرئيسية هناك دوال أخري مثل grid سنشرحها في حين الوصول اليه.**
3. **الأسطر من 9 إلى 20 قمنا بتعريف الأزرار المطلوبة وإعطائها نص من خلال ملف constantes الذي أنشأناه مسبقا**
4. **لاحظ كل العناصر يجب أن تُعرف (تكتب) قبل استدعاء دالة mainloop() لكي تضاف إلي الشاشة.**

قم بتشغيل البرنامج الأن



ستظهر العناصر على الشاشة الرئيسية ولكنها غير مرتبة.

دالة grid وترتيب العناصر

الـgrid تقوم بتقسيم الشاشة إلي عدد من الأعمدة والصفوف
الأعمدة مرتبة من اليسار لليمين وتبدأ 1و2و3و4.. وهكذا
الصفوف مرتبة من اليسار لليمين وتبدأ من 1و2و3و4... وهكذا
موضع العنصر هو تقاطع المحور الراسي مع الأفقي.

```
grid(row=0,column=0)
```

تكتب دالة grid هكذا

حيث row تمثل المحور الرئيسي الصفوف وcolumn تمثل المحور الأفقي العمود في المثال السابق الصف الأول العمود الأول صفر تعني استخدام كامل مساحة الصف أو العمود دون تقسيمة.

لو قيمة ال row أو ال column صفر يعني تحديد الصف كله أو العمود

إعادة ترتيب العناصر في الشاشة

```
#Zakat Calculator
from tkinter import *
from tkinter import ttk
import constantes as define

window = Tk()
window.title(define.PROGTITLE)

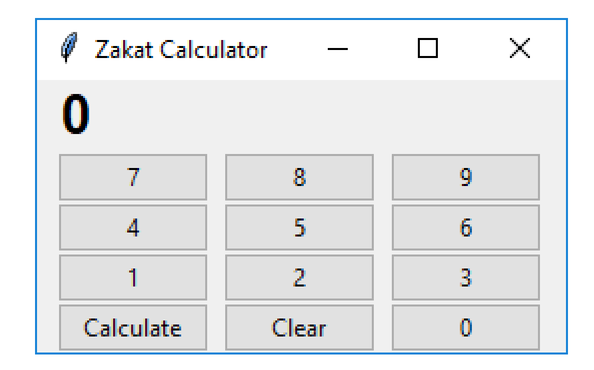
#first row
screen_label = ttk.Label(text="0",font=("Arial", 20,"bold"),width=16).grid(row=0,column=0,columnspan=5,padx=10)
#second row btn 987
btn_sevin = ttk.Button(text=define.BUTTONSEVIEN).grid(row=1,column=1)
btn_egiht = ttk.Button(text=define.BUTTONEIGHT).grid(row=1,column=2)
btn_nine = ttk.Button(text=define.BUTTONNINE).grid(row=1,column=3)

#Three row 654
btn_four = ttk.Button(text=define.BUTTONFOUR).grid(row=2,column=1)
btn_five = ttk.Button(text=define.BUTTONFIVE).grid(row=2,column=2)
btn_six = ttk.Button(text=define.BUTTONSIX).grid(row=2,column=3)

#forty row 123
btn_one = ttk.Button(text=define.BUTTONONE).grid(row=3,column=1)
btn_tow = ttk.Button(text=define.BUTTONTOW).grid(row=3,column=2)
btn_three = ttk.Button(text=define.BUTTONTHREE).grid(row=3,column=3)

#fifty row 0 calc clear
btn_calcu = ttk.Button(text=define.BUTTONCALTEXT).grid(row=4,column=1)
btn_clear = ttk.Button(text=define.BUTTONCLEAR).grid(row=4,column=2)
btn_zero = ttk.Button(text=define.BUTTONZERO).grid(row=4,column=3)
window.mainloop(
```

قم بتشغيل البرنامج



الأن لدينا برنامج مرتب ورائع

المرحلة الثانية ال action

في هذه المرحلة سوف نقوم بربط كل عنصر بالفعل الخاص به كزر حساب الزكاة كأنك تسأل نفسك ما الذي سيحدث عند الضغط علية.

سنحتاج إلي ثلاث دوال رئيسية الأولي مع الزر **clear** لإعادة تهيئة الشاشة والثانية مع **Calculate** لحساب الزكاة والثالثة مع الأرقام لتمرير قيمة كل رقم.

> ملحوظة لتنفيذ الأمر على العنصر لابد من استخدام **parameter** **command**بإعطائه اسم الدالة فقط دون أي متغيرات ولكن ماذا لو أردت استخدام دالة تحتوي على متغيرات نستخدم دالة **lambda** لتجاوز هذه المشكلة

الأن سنقوم بتعريف الدوال

```
#Zakat Calculator
"""actions start"""
def do_clear():
    global resultString
    resultString.set(0)

def do_calculat():
    global resultString
    amount = resultString.get()
    amount = int(amount)
    if amount < 100:
        resultString.set("0")
    else:
        zakat = float(amount)/40
        resultString.set(zakat)

def add_number(number):
    global resultString
    if resultString.get() == "0":
        screen_ammonut = str(number)
    else:
        screen_ammonut = resultString.get()+str(number)

    resultString.set(screen_ammonut)
"""actions end"""

resultString = StringVar()
resultString.set(0)

#first row
screen_label = ttk.Label(text="0",textvariable=resultString,font=("Arial",20,"bold"),width=16).grid(row=0,column=0,columnspan=5,padx=10)

```

1. السطر 3 قمنا بتعريف دالة do_clear وهي المسئولة عن إعادة تهيئة العملية الحسابية وتصفير البرنامج.
2. السطر 4 استدعينا متغير خارجي resultString الذي قمنا بتعريفة في السطر 26 وأعطيناه قيمة افتراضية في السطر 27 ثم أضفناه إلي الـlabel في السطر 30 textvariable=resultString هذا المتغير من نوع string هو المتغير المسئول عن إظهار القيمة في الـlabel.
3. السطر 5 قمنا بإعطاء المتغير قيمة 0 لإظهارها في الـlabel.
4. السطر 7 قمنا بتعريف دالة do_calculat لحساب نسبة الزكاة.
5. السطر 8 قمنا باستدعاء متغير قيمة الـ label من خارج الدالة.
6. السطر 9 أعطينا المتغير amount القيمة الموجودة في label وهو المبلغ واجب الزكاة المدخل من قبل المستخدم عن طريق دالة get.
7. السطر 10 حولنا amount من string إلي int لان المتغير resultString هو في الأصل string.
8. السطر 11 أضفنا الشرط لو المبلغ اقل من 100 وهو نصاب الزكاة إذن لا تجب الزكاة ليكون مخرجات الدالة صفر (راجع مرحلة التحليل).
9. السطر 14 قسمنا المبلغ على 40 هو نسبة زكاة المال (راجع مرحلة التحليل) قمنا جعل الرقم بنوع float لاحتمال أن يكون الناتج به أرقام عشرية.
10. السطر 15 حدثنا ال label (الشاشة)بقيمة الزكاة عن طريق دالة set.
11. السطر 17 عرفنا دالة add_number وأعطيناها متغير number وهي الدالة المسئولة عن إضافة الرقم الموجود على الزر عند الضغط علية إلي الشاشة.
12. السطر 18 استدعينا متغير resultString من خارج الدالة.
13. السطر 19 قمنا بإضافة شرط لو أن القيمة المجلبة من الشاشة = صفر سنقوم باستبدالها بقيمة الزر.
14. السطر 20 عرفنا متغير screen_ammonut وأعطيناه قيمة الزر في حالة أن الرقم في الشاشة 0.

15.السطر 22 قمنا بإضافة قيمة الزر إلي جوار الرقم الموجود (إضافتها وليس جمعها) لاحظ أن متغيرnumber غيرنا نوعه إلي string باستخدام دالة str لان label (الشاشة)يقبل string فقط.

16.السطر 24 أعطينا متغير resultString ناتج عملية القسمة باستخدام دالة set.

لنقم الأن بإضافة الدوال إلي الأزرار عن طريق متغير

```
command=lambda: function_name(params)
```

```
#second  row btn 987
btn_sevin = ttk.Button(text=define.BUTTONSEVIEN,command=lambda:add_number(7)).grid(row=1,column=1)
btn_egiht = ttk.Button(text=define.BUTTONEIGHT,command=lambda:add_number(8)).grid(row=1,column=2)
btn_nine = ttk.Button(text=define.BUTTONNINE,command=lambda:add_number(9)).grid(row=1,column=3)

#Three row 654
btn_four = ttk.Button(text=define.BUTTONFOUR,command=lambda:add_number(4)).grid(row=2,column=1)
btn_five = ttk.Button(text=define.BUTTONFIVE,command=lambda:add_number(5)).grid(row=2,column=2)
btn_six = ttk.Button(text=define.BUTTONSIX,command=lambda:add_number(6)).grid(row=2,column=3)

#forty row 123
btn_one = ttk.Button(text=define.BUTTONONE,command=lambda:add_number(1)).grid(row=3,column=1)
btn_tow = ttk.Button(text=define.BUTTONTOW,command=lambda:add_number(2)).grid(row=3,column=2)
btn_three = ttk.Button(text=define.BUTTONTHREE,command=lambda:add_number(3)).grid(row=3,column=3)

#fifty  row 0 cal clear
btn_calcu = ttk.Button(text=define.BUTTONCALTEXT,command=lambda:do_calculat()).grid(row=4,column=1)
btn_clear = ttk.Button(text=define.BUTTONCLEAR,command=lambda:do_clear()).grid(row=4,column=2)
btn_zero = ttk.Button(text=define.BUTTONZERO,command=lambda:add_number(0)).grid(row=4,column=3)
```

قم بتشغيل برنامج الأن واحسب زكاتك الواجبة

Zakat Calculator

12000

9	8	7
7	5	4
3	2	1
0	Calculate	Clear

Zakat Calculator

300.0

9	8	7
7	5	4
3	2	1
0	Calculate	Clear

قم بعمل زر التراجع يستخدم لحذف أخر رقم موجود في **label** في كل مرة يتم الضغط علية

للحصول على الكود كامل من هذا الرابط
https://github.com/alnazer/Zakat-Calculator

المرحلة الثالثة التصدير

في هذه المرحلة سوف نصدر برنامجنا بامتداد **exe** ليأخذ خصائص البرامج العادية يمكنك تسويقه وتشغيله دون الحاجة إلي برنامج **eclipse** بعد الأن.

أولا تهيئة بيئة العمل

قم بتثبيت مكتبة pyinstaller من خلال command windows

أولا قم بتعريف مسار البايثون على الويندوز من خلال اتباع الخطوات التالية

1- من سطح المكتب اضغط بالزر الأيمن للفارة على خصائص (**proparites**) ستظهر لك نافذة بخصائص الجهاز من اليسار اختر **advanced system setting**

2- ستظهر لك نافذة اختر منها **Environment Variables**

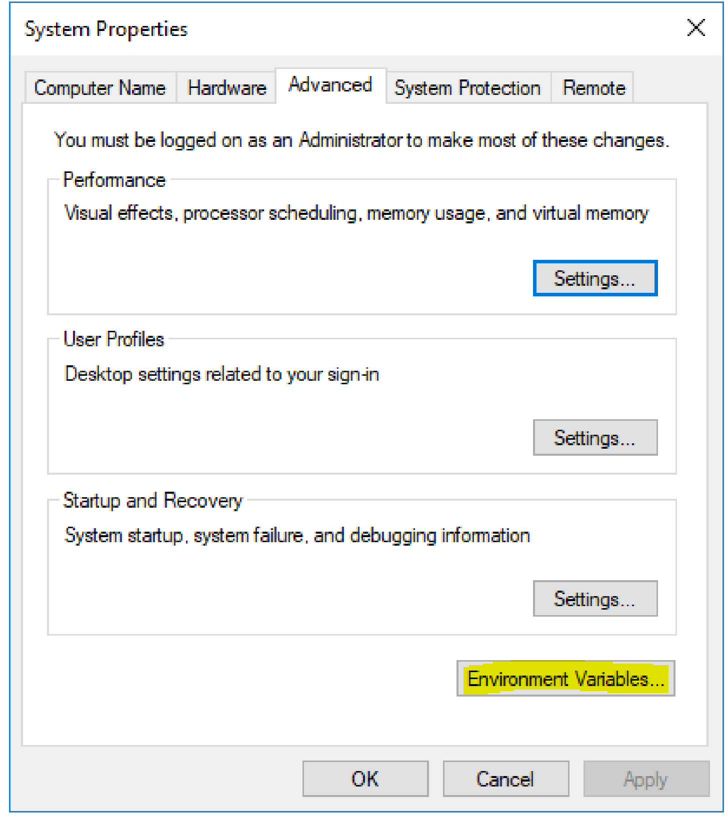
Control Panel Home

Device Manager

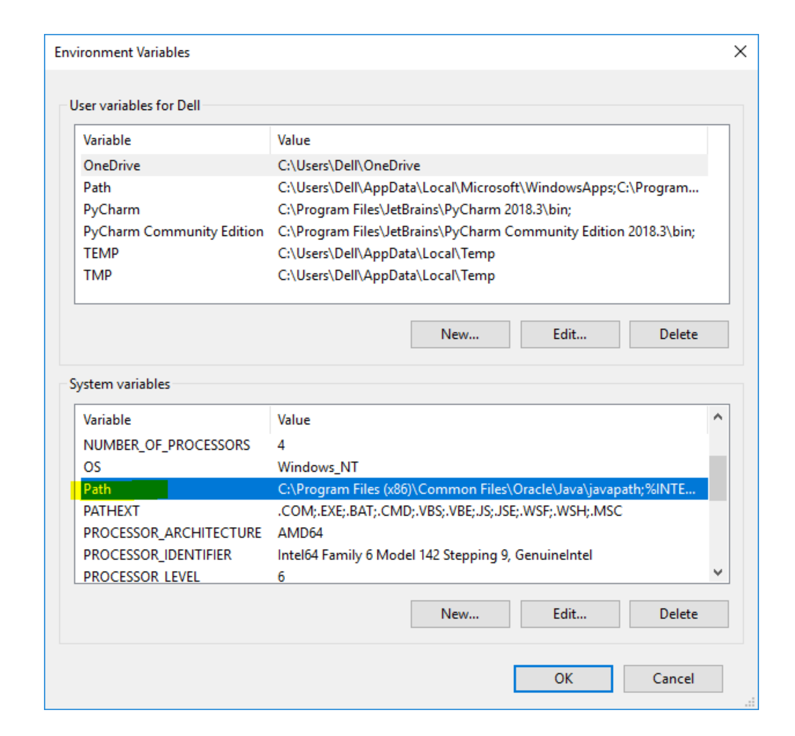
Remote settings

System protection

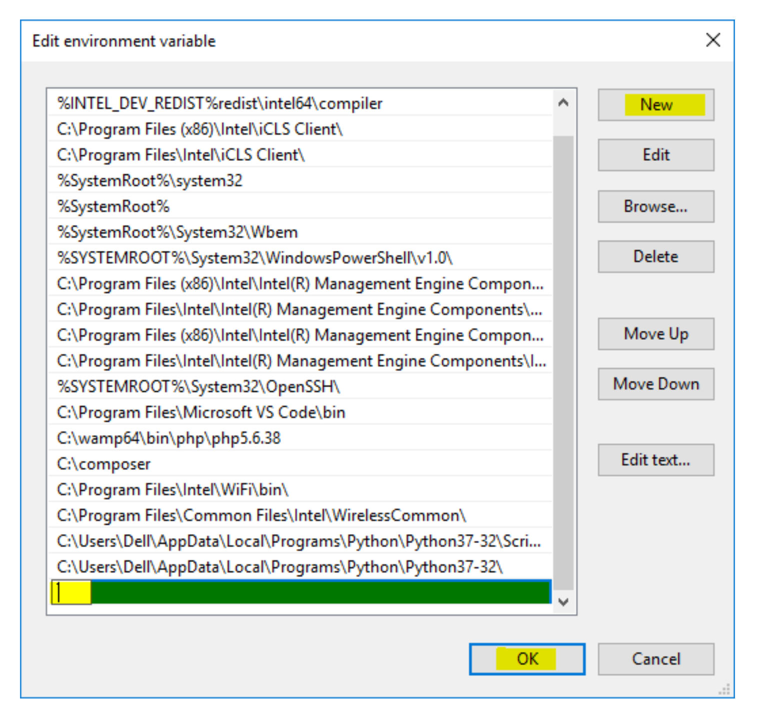
Advanced system settings



3- من المربع الثاني حدد كلمة **path** ثم اضغط **edit**



4- ستظهر لك نافذة اضغط على new



5- سيتم إضافة سطر جديد في اخر القائمة قم بلصق مسار برنامج البايثون الذي قمنا بتثبيتة مسبقا في الدروس الاولي من الكتاب مثال **C:\Users\Dell\AppData\Local\Programs\Python\Python37-32\Scripts** لابد ان يكون مجلد**Scripts** موجود وتنتهي بشرطة مائله \

6- اظف سطر اخر واضف فية مثار البايثون **C:\Users\Dell\AppData\Local\Programs\Python\Python37-32** لابد ان ينتهي ب \

المسارات السابقة تختلف من جهاز الي اخر ان لم تستطع إيجاد المسار يمكنك البحث عن مجلد Python وتحديد المسار من نتائج البحث

لنتأكد الأن من نجاحنا فب تعريف بايثون افتح **command**الخاصة بالويندوز من خلال التوجه إلى قائمة **start**والبحث عن كلمة **command** أو التوجه إلي المسار.
C:\WINDOWS\system32\cmd.exe

ثم نكتب كلمة python ونضغط زر enter من لوحة المفاتيح (سيظهر لك نسخة البايثون **3.7.1**)

```
Command Prompt - python

C:\Users\Dell>python
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

تثبيت مكتبة pip بكتابة الأمر python -m pip install –U pip

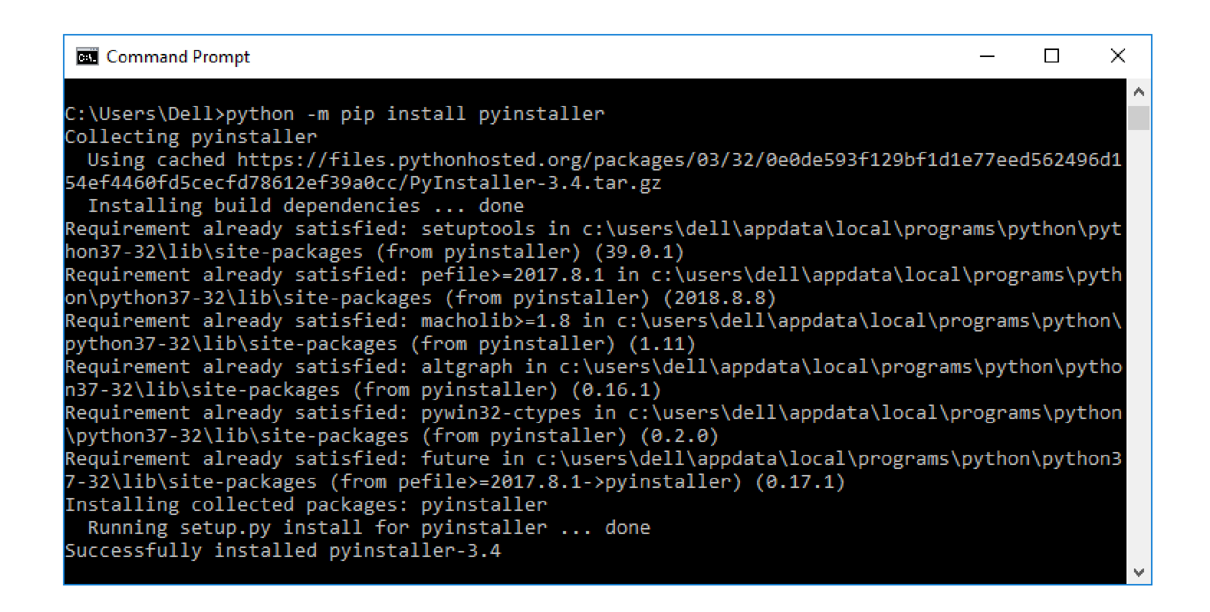
```
Command Prompt

Microsoft Windows [Version 10.0.17134.407]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Dell>python -m pip install -U pip
Collecting pip
  Using cached https://files.pythonhosted.org/packages/c2/d7/90f34cb0d83a6c5631cf71df
e64cc1054598c843a92b400e55675cc2ac37/pip-18.1-py2.py3-none-any.whl
Installing collected packages: pip
  Found existing installation: pip 10.0.1
    Uninstalling pip-10.0.1:
      Successfully uninstalled pip-10.0.1
Successfully installed pip-18.1
```

لتثبيت مكتبة pyinstaller نكتب الأمر التالي

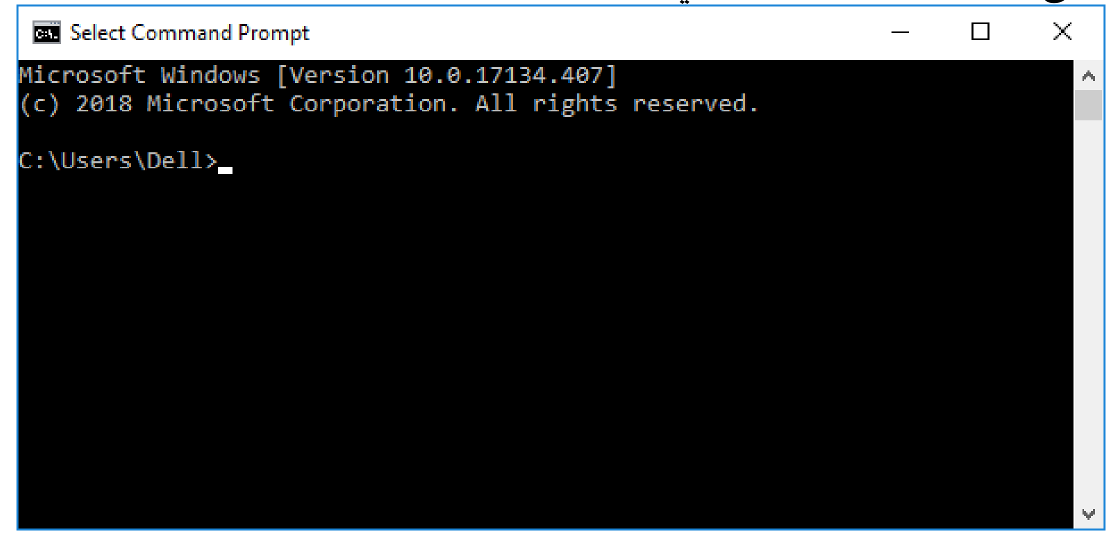
python –m pip install pyinstaller



```
C:\Users\Dell>python -m pip install pyinstaller
Collecting pyinstaller
  Using cached https://files.pythonhosted.org/packages/03/32/0e0de593f129bf1d1e77eed562496d1
54ef4460fd5cecfd78612ef39a0cc/PyInstaller-3.4.tar.gz
  Installing build dependencies ... done
Requirement already satisfied: setuptools in c:\users\dell\appdata\local\programs\python\pyt
hon37-32\lib\site-packages (from pyinstaller) (39.0.1)
Requirement already satisfied: pefile>=2017.8.1 in c:\users\dell\appdata\local\programs\pyth
on\python37-32\lib\site-packages (from pyinstaller) (2018.8.8)
Requirement already satisfied: macholib>=1.8 in c:\users\dell\appdata\local\programs\python\
python37-32\lib\site-packages (from pyinstaller) (1.11)
Requirement already satisfied: altgraph in c:\users\dell\appdata\local\programs\python\pytho
n37-32\lib\site-packages (from pyinstaller) (0.16.1)
Requirement already satisfied: pywin32-ctypes in c:\users\dell\appdata\local\programs\python
\python37-32\lib\site-packages (from pyinstaller) (0.2.0)
Requirement already satisfied: future in c:\users\dell\appdata\local\programs\python\python3
7-32\lib\site-packages (from pefile>=2017.8.1->pyinstaller) (0.17.1)
Installing collected packages: pyinstaller
  Running setup.py install for pyinstaller ... done
Successfully installed pyinstaller-3.4
```

ثانيا التصدير النهائي

افتح نافذة الأوامر بالذهاب إلي قائمة start وابحث عن command



```
Microsoft Windows [Version 10.0.17134.407]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Dell>_
```

لتحويل ملفات البايثون إلي برنامج exe اكتب الأمر التالي

```
Command Prompt
Microsoft Windows [Version 10.0.17134.407]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Dell>pyinstaller -w -F C:\Users\Dell\eclipse-workspace\zakat\main.py
```

امر تصدير البرنامج — مسار ملف البرنامج علي الجهاز

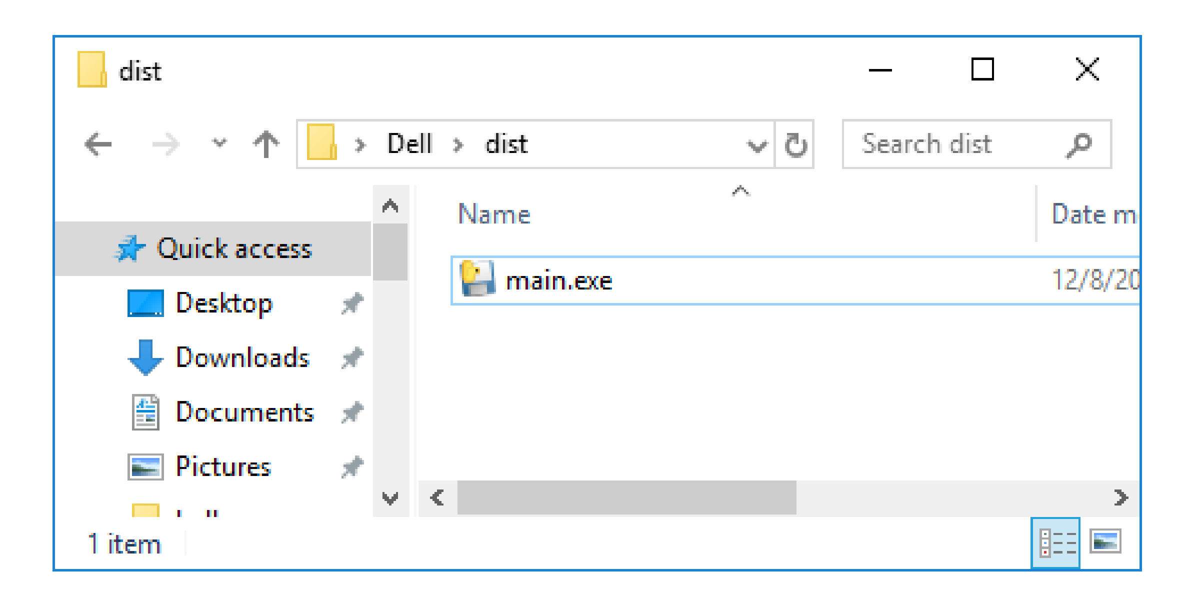
pyinstaller -w -F C:\Users\Dell\eclipse-workspace\zakat\main.py

في النهاية سيخبرك مكان حفظ الملف في الجهاز

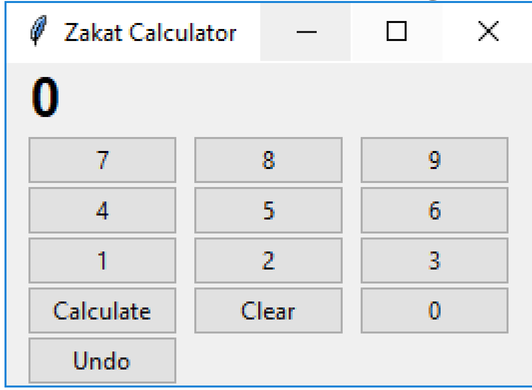
```
Command Prompt
67 INFO: checking Analysis
157 INFO: checking PYZ
170 INFO: checking PKG
247 INFO: Bootloader C:\Users\Dell\AppData\Local\Programs\Python\Python37-32\lib\site-pa
ckages\PyInstaller\bootloader\Windows-32bit\runw.exe
247 INFO: checking EXE
256 INFO: Rebuilding EXE-00.toc because main.exe missing
256 INFO: Building EXE from EXE-00.toc
257 INFO: Appending archive to EXE C:\Users\Dell\dist\main.exe
375 INFO: Building EXE from EXE-00.toc completed successfully.

C:\Users\Dell>
```

توجه إلي المسار المذكور الأن أصبح لديك برنامجك الخاص



قم بفتح البرنامج بالضغط علية ضغطتين بزر الفارة الأيسر



مبروك أصبح لديك برنامجك الخاص بإمكانك الأن نشرة على الأنترنت
لتغيير أيقونة البرنامج بدلا من الأيقونة الافتراضية نستخدم امر I- يليه مسار الأيقونة

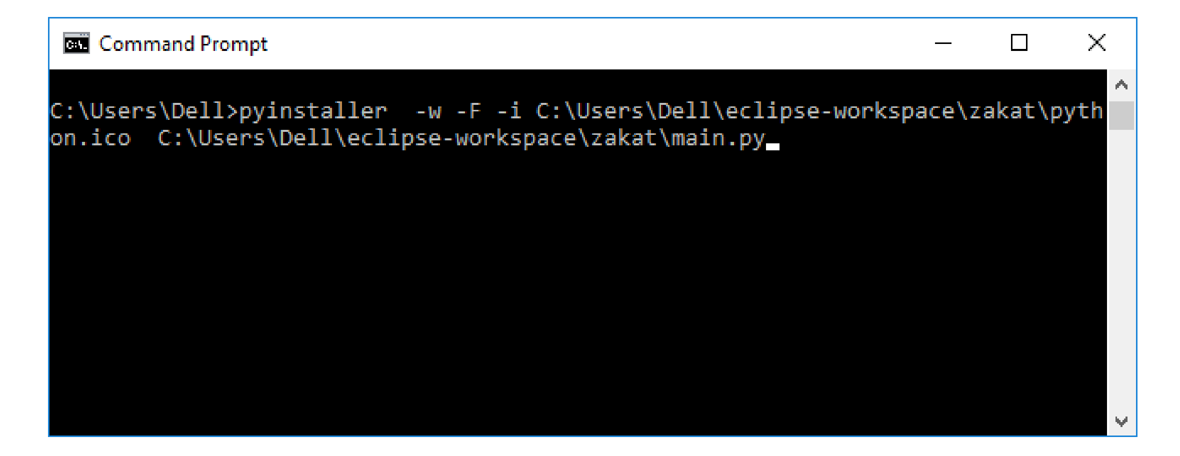
الامتداد المسموح به للأيقونة هو ico

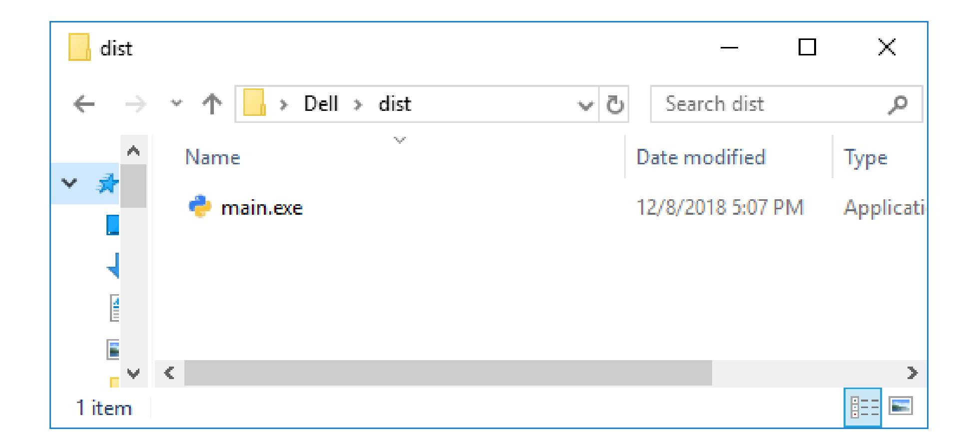
حمل الأيقونة من هنا

https://raw.githubusercontent.com/alnazer/Zakat-Calculator/master/python.ico

انسخها وألصقها في مجلد البرنامج ونفذ الأمر

pyinstaller -w -F -i C:\Users\Dell\eclipse-workspace\zakat\python.ico C:\Users\Dell\eclipse-workspace\zakat\main.py





كل ملفات البرنامج (المثال التطبيقي + الملفات الأساسية) متاحه للتحميل على هذا الرابط https://github.com/alnazer/Zakat-Calculator

الخاتمة

تم بحمد الله الانتهاء من شرح أساسيات لغة البايثون مع التطبيق على مثال عملي (**برنامج حاسبة الزكاة**).
هناك بعض الأسئلة التي قد تتوارد في ذهنك دعنا نتوقعها ونجيب عليها.

1. **هل أصبحت الأن مبرمج لغة بايثون؟**

نعم أنت الأن مبرمج لغة بايثون ولكن مبرمج مبتدئي فلا يمكنك العمل على مشاريع كبيرة مازال الطريق أمامك للتعلم لكنك قطعت شوط كبير جدا فيه.

2. **متي ادن أصبح مبرمج بايثون محترف؟**

هل معني استخراجك رخصة قيادة أنك تستطيع قيادة قاطرة بالتأكيد لا. لكي تصبح مبرمج محترف يجب عليك معرفة python object oriented أو ما نسميه في المنطقة العربية كلاسيز ثم التخصص في مجال معين من مجالات بايثون.

3. **هل هناك مجالات كثيرة لبايثون؟**

نعم يوجد العديد من مجالات (التخصصات) في بايثون كبرمجة الألعاب الثانية البعد والثلاثية البعد برمجة الاله وبرمجة الويب وبرمجة برامج سطح المكتب عليك أن تختار وتقرر.

4. **هل معني تخصصي في تخصص أو مجال معين تعلُم لغة جديدة؟**

إطلاقاً كل تلك المجالات مبنية على مكتبات أو الframeworks المبينة على لغة بايثون عليك فقط معرفة ماهية عمل المكتبة (هل تتذكر مكتبة tkinter) ولكن لا تنسي أن كل هذه المكتبان مبنية على python object oriented.

5. **ماذا على أن افعل الأن؟**

اعلم تمام العلم أن الكتب والكورسات لن تجعلك مبرمج محترف هي فقط تضعك على أول الطريق (كمن يدلك على هدفك ويعطيك زاد الطريق) اذهب إلى فضاء الأنترنت وحاول مراجعة اكواد مبرمجين محترفين وفهمها بناءً على ما تعلمت هذه الطريقة ستكسبك خبرات كبيرة وستجعلك تفكر كالمبرمجين المحترفين.

كلمة المؤلف



96590033807

hassanaliksa@gmail.com

https://www.linkedin.com/in/hassan-ali-2310905b

PHP
Java
Android
Python
Yii 2.0 framework
CSS 3
Photoshop
Json,Xml,jQuery
Html5
REST API (Json-Xml)
Bootstrap
WordPress

عزيزي مبرمج المستقبل

أتمنى من الله العلى القدير أن ينال هذا الكتاب إعجابك فقد حاولت بقدر المستطاع أن أتناول الشرح بالتبسيط وببعض الأمثلة التوضيحية وأن ابتعد عن التفاصيل المملة التي لا تزيد الأمر إلا تعقيدا وحشواً لا فائدة منه ودعمت شرحي ببعض الأمثلة وحرصت على أن اشرح كل نقطة قد تسبب لك لبسا أو صعوبة في الفهم.

اعلم أخي المبرجم أن الفهم من الله وحده ﴿فَفَهَّمْنَاهَا سُلَيْمَانَ﴾ فكن صادق النية مع الله ولا تتعالي بعلمك على أحد أو تكتم يوما علماً قال النبي ﷺ: من سئل عن علم فكتمه ألجمه الله يوم القيامة بلجام من نار، فكن عوننا لغيرك.

واعلم أن للمبرمج الناجح صفات يجب أن يتحلا بها أولهما الصبر المدعوم بالمثابرة في معرفة الشيء والشغف والمجاهدة للحصول على المعلومة ولم يكن المبرمج الناجح ابدآ متعالي على طلب المعرفة.

حاول ألا تكون منعزلاً عن عالمك انفتح على أصدقائك ولا تلتفت إلى مخذل أو محبط.

أن ثقتك بنفسك هي أول طريقك للنجاح (نعم تستطيع أن تنجح فليس الناجحون اقل منك)

فبدا الأن......

آخوك حسن على

جدول المحتويات



اتمني من الله العلى القدير ان ينال هذا الكتاب اعجابكم فقد حاولت بقدر المستطاع ان اتناول الشرح بالتبسيط وببعض الأمثلة التوضيحية وان ابتعد عن التفاصيل المملة التي لا تزيد الامر الا تعقيدا وحشواً لا فائدة منه ودعمت شرحي ببعض الأمثلة وحرصت على ان اشرح كل نقطة قد تسبب لكم لبسا او صعوبة في الفهم

اخوكم حسن علي

رقم ايداع 2019/212/12554